Randomized greedy inference for joint segmentation, POS tagging and dependency parsing

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.


As Published: http://dblp.dagstuhl.de/db/conf/naacl/naacl2015.html

Publisher: Association for Computational Linguistics

Persistent URL: http://hdl.handle.net/1721.1/110740

Version: Author’s final manuscript: final author’s manuscript post peer review, without publisher’s formatting or copy editing

Terms of use: Creative Commons Attribution-Noncommercial-Share Alike
Randomized Greedy Inference for Joint Segmentation, POS Tagging and Dependency Parsing

Yuan Zhang, Chengtao Li, Regina Barzilay
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{yuanzh, ctli, regina}@csail.mit.edu

Kareem Darwish
ALT Research Group
Qatar Computing Research Institute
kdarwish@qf.org.qa

Abstract

In this paper, we introduce a new approach for joint segmentation, POS tagging and dependency parsing. While joint modeling of these tasks addresses the issue of error propagation inherent in traditional pipeline architectures, it also complicates the inference task. Past research has addressed this challenge by placing constraints on the scoring function. In contrast, we propose an approach that can handle arbitrarily complex scoring functions. Specifically, we employ a randomized greedy algorithm that jointly predicts segmentations, POS tags and dependency trees. Moreover, this architecture readily handles different segmentation tasks, such as morphological segmentation for Arabic and word segmentation for Chinese. The joint model outperforms the state-of-the-art systems on three datasets, obtaining 2.1% TedEval absolute gain against the best published results in the 2013 SPMRL shared task.1

1 Introduction

Parsing accuracy is greatly impacted by the quality of preprocessing steps such as tagging and word segmentation. Li et al. (2011) report that the difference between using the gold POS tags and using the automatic counterparts reaches about 6% in dependency accuracy. Prior research has demonstrated that joint prediction alleviates error propagation inherent in pipeline architectures, where mistakes cascade from one task to the next (Bohnet et al., 2013; Tratz, 2013; Hatori et al., 2012; Zhang et al., 2014a). However, jointly modeling all the processing tasks inevitably increases inference complexity. Prior work addressed this challenge by introducing constraints on scoring functions to keep inference tractable (Qian and Liu, 2012).

In this paper, we propose a method for joint prediction that imposes no constraints on the scoring function. The method is able to handle high-order and global features for each individual task (e.g., parsing), as well as features that capture interactions between tasks. The algorithm achieves this flexibility by operating over full assignments that specify segmentation, POS tags and dependency tree, moving from one complete configuration to another.

Our approach is based on the randomized greedy algorithm from our earlier dependency parsing system (Zhang et al., 2014b). We extend this algorithm to jointly predict the segmentation and the POS tags in addition to the dependency parse. The search space for the algorithm is a combination of parse trees and lattices that encode alternative morphological and POS analyses. The inference algorithm greedily searches over this space, iteratively making local modifications to POS tags and dependency trees. To overcome local optima, we employ multiple restarts.

This simple, yet powerful approach can be easily applied to a range of joint prediction tasks. In prior work, joint models have been designed for a specific language. For instance, joint models for Chinese are designed with word segmentation in mind (Hatori et al., 2012), while algorithms for processing Semitic languages are tailored for morpho-

---

1The source code is available at https://github.com/yuanzh/SegParser.
logical analysis (Tratz, 2013; Goldberg and Elhadad, 2011). In contrast, we show that our algorithm can be effortlessly applied to all these distinct languages. Language-specific characteristics drive the lattice construction and the feature selection, while the learning and inference methods are language-agnostic.

We evaluate our model on three datasets: SPMRL (Modern Standard Arabic), classical Arabic and CTB5 (Chinese). Our model consistently outperforms state-of-the-art systems designed for these languages. We obtain a 2.1% TedEval gain against the best published results in the 2013 SPMRL shared task (Seddah et al., 2013). The joint model results in significant gains against its pipeline counterpart, yielding 2.4% absolute F-score increase in dependency parsing on the same dataset. Our analysis reveals that most of this gain comes from the improved prediction on OOV words.

2 Related Work

Joint Segmentation, POS tagging and Syntactic Parsing It has been widely recognized that joint prediction is an appealing alternative for pipeline architectures (Goldberg and Tsarfaty, 2008; Hatori et al., 2012; Habash and Rambow, 2005; Gahbiche-Braham et al., 2012; Zhang and Clark, 2008; Bohnet and Nivre, 2012). These approaches have been particularly prominent for languages with difficult pre-processing, such as morphologically rich languages (e.g., Arabic) and languages that require word segmentation (e.g., Chinese). For the former, joint prediction models typically rely on a lattice structure to represent alternative morphological analyses (Goldberg and Tsarfaty, 2008; Tratz, 2013; Cohen and Smith, 2007). For instance, transition-based models intertwine operations on the lattice with operations on a dependency tree. Other joint architectures are more decoupled: in Goldberg and Tsarfaty (2008), a lattice is used to derive the best morphological analysis for each part-of-speech alternative, which is in turn provided to the parsing algorithm. Our analysis shows how to apply this strategy to a more challenging inference task and demonstrate that a randomized greedy algorithm achieves excellent performance in a significantly larger search space.

3 Randomized Greedy System for Joint Prediction

In this section, we introduce our model for joint morphological segmentation, tagging and parsing. Our description will first assume that word boundaries are provided (e.g., the case of Arabic). Later, we will describe how this model can be applied to a joint prediction task that involves word segmentation (e.g., Chinese).

3.1 Notation

Let \( x = \{x_i\}_{i=1}^{|x|} \) be a sentence of length \(|x|\) that consists of tokens \( x_i \). We use \( s = \{s_i\}_{i=1}^{|x|} \) to denote a segmentation of all the tokens in sentence \( x \), and \( s_i = \{s_{i,j}\}_{j=1}^{|s_i|} \) to denote a segmentation of the token \( x_i \), where \( s_{i,j} \) is the \( j \)th morpheme of the token \( x_i \). Similarly, we use \( t \), \( t_i \) and \( t_{i,j} \) for the POS
tags for each sentence, token and morpheme. We use $y$ to denote a dependency tree over morphemes, and $y_{i,j}$ to denote the head of morpheme $s_{i,j}$. During training, the algorithm is provided with tuples that specify ground truth values for all the variables $\mathcal{D} = \{(x, s, t, y)\}$.

We also assume access to a morphological analyzer and a POS tagger that provide candidate analyses. Specifically, for each token $x_i$, the algorithm is provided with candidate segmentations $S_i$, and candidate POS tags $T_i$ and $T_{i,j}$. These alternative analyses are captured in the lattice structure (see Figure 1 for an example). Finally, we use $\mathcal{Y}$ to denote the set of all valid dependency trees defined over morphemes.

### 3.2 Decoding

We parameterize the scoring function as

$$score(x, s, t, y) = \theta \cdot f(x, s, t, y)$$

(1)

where $\theta$ is the parameter vector and $f(x, s, t, y)$ is the feature vector associated with the sentence and all variables.

The goal of decoding is to find a set of valid values $(s, t, y) \in S \times T \times \mathcal{Y}$ that maximizes the score defined in Eq. 1. Our randomized greedy algorithm finds a high scoring assignment for $(s, t, y)$ via a hill-climbing process with multiple random restarts. (Section 3.3 describes how the parameters $\theta$ are learned.)

Figure 2 shows the framework of our randomized greedy algorithm. First, we draw a full path from the lattice structure in two steps: (1) sampling a morphological segmentation $s$ from $S$; (2) sampling POS tags $t$ for each morpheme. Next, we sample a dependency tree $y$ from the parse space. Based on this random starting point, we iteratively hill-climb $t$ and $y$ in a bottom-up order.\(^2\) In our earlier work (Zhang et al., 2014b), we showed this strategy guarantees that we can climb to any target tree in a finite number of steps. We repeat the sampling and the hill-climbing processes above until we do not find a better solution for $K$ iterations. We introduce the details of this process below.

**SampleSeg and SamplePOS:** Given a sentence $x$, we first draw segmentations $s$ and POS tags $t^{(0)}$ from the first-order distribution using the current learned parameter values. For segmentation, first-order features only depend on each token $x_i$ and its morphemes $s_{i,j}$. Similarly, for POS, first-order features are defined based on $s_{i,j}$ and $t_{i,j}$. The sampling process is straightforward due to the fact that the candidate sets $|S|$ and $|T|$ are both small. We can enumerate and compute the probabilities proportional to the exponential of the first-order scores as follows.$\(^3\)

$$p(s_i) \propto \exp\{\theta \cdot f(x, s_i)\}$$

$$p(t_{i,j}) \propto \exp\{\theta \cdot f(x, s_i, t_{i,j})\}$$

(2)

**SampleTree:** Given a random sample of the segmentations $s$ and the POS tags $t^{(0)}$, we draw a random tree $y^{(0)}$ from the first-order distribution using Wilson’s algorithm (Wilson, 1996).\(^4\)

**HillClimbPOS:** After sampling the initial values $s, t^{(0)}$ and $y^{(0)}$, the hill-climbing algorithm improves the solution via locally greedy changes. The hill-climbing algorithm iterates between improving the POS tags and the dependency tree. For POS tagging, it updates each $t_{i,j}$ in a bottom-up order as follows

$$t_{i,j} \leftarrow \arg \max_{t_{i,j} \in T_{i,j}} score(x, s, t_{i,j}, t_{-(i,j)}, y)$$

(3)

where $t_{-(i,j)}$ are the rest of the POS tags when we update $t_{i,j}$.

---

\(^2\)We do not hill-climb segmentation, or else we have to jointly find the optimal $t$ and $y$, and the resulting computational cost is too high.

\(^3\)We notice that the distribution becomes significantly sharper after training for several epochs. Therefore, we also smooth the distribution by multiplying the score with a scaling factor.

\(^4\)We also smooth the distribution in the same way as in segmentation and POS tagging.
**Input:** parameter θ, sentence x

**Output:** segmentations s, POS tags t and dependency tree y

1: s ← SampleSeg(x)
2: t(0) ← SamplePos(x, s)
3: y(0) ← SampleTree(x, s, t(0))
4: k = 0
5: repeat
6:   t(k+1) ← HillClimbPOS(x, s, t(k), y(k))
7:   y(k+1) ← HillClimbTree(x, s, t(k+1), y(k))
8:   k ← k + 1
9: until no change in this iteration
10: return (s, t(k), y(k))

Figure 2: The hill-climbing algorithm with random initializations. Details of the sampling and hill-climbing functions in Line 1-3 and 6-7 are provided in Section 3.2.

**HillClimbTree:** We improve the dependency tree y via a similar hill-climbing process. Specifically, we greedily update the head y_{i,j} of each morpheme in a bottom-up order as follows

\[ y_{i,j} \leftarrow \arg \max_{y_{i,j} \in \mathcal{Y}_{i,j}} \text{score}(x, s, t, y_{i,j}, y_{-(i,j)}) \]  

(4)

where \( \mathcal{Y}_{i,j} \) is the set of candidate heads such that changing \( y_{i,j} \) to any candidate does not violate the tree constraint.

### 3.3 Training

We learn the parameters θ in a max-margin framework, using on-line updates. For each update, we need to compute the segmentations, POS tags and the tree that maximize the cost-augmented score:

\[
(s, \tilde{t}, \tilde{y}) = \arg \max_{s \in \mathcal{S}, t \in \mathcal{T}, y \in \mathcal{Y}} \{\theta \cdot f(x, s, t, y) + Err(s, t, y)\}
\]  

(5)

where \( Err(s, t, y) \) is the number of errors of \((s, t, y)\) against the ground truth \((\hat{s}, \hat{t}, \hat{y})\). The parameters are then updated to guide the selection against the violation. This is done via standard passive-aggressive updates (Crammer et al., 2006).

### 3.4 Adapting to Chinese Joint Prediction

In this section we describe how the proposed model can be adapted to languages that do not delineate words with spaces, and thus require word segmentation. The main difference lies in the construction of the lattice structure. We employ a state-of-the-art word segmenter to produce candidate word boundaries. We consider boundaries common across all the top-k candidates as true word boundaries. The remaining tokens (i.e., strings between these boundaries) are treated as words to be further segmented and labeled with POS tags. Figure 3 shows an example of the Chinese word lattice structure we construct. Once the lattice is constructed, the joint prediction model is applied as described above.

### 4 Features

#### Segmentation Features

For both Arabic and Chinese, each segmentation is represented by its score from the preprocessing system, and by the corresponding morphemes (or words in Chinese). Following previous work (Zhang and Clark, 2010), we also add character-based features for Chinese word segmentation, including the first and the last characters in the word, and the length of the word.

#### POS Tag Features

Table 1 summarizes the POS tag features employed by the model. First, we use the feature templates proposed in our previous work on Arabic joint parsing and POS correction (Zhang et al., 2014c). In addition, we incorporate character-based features specifically designed for Chinese. These features are mainly inspired by previous transition-based models on Chinese joint POS tagging and word segmentation (Zhang and Clark, 2010).

#### Dependency Parsing Features

The feature templates for dependency parsing are mainly drawn from our previous work (Zhang et al., 2014b). Fig-

Figure 3: Example lattice structures for the Chinese sentence “新华 社 北京 二月 十三 日 报告” (Xinhua Press at Beijing reports on February 13th). The token 新华 社 has two candidate segmentations: 新华 社 or 新华 + 社.
Table 1: POS tag feature templates. $t_0$ and $w_0$ denote the POS tag and the word at the current position. $t_{-x}$ and $t_2$ denote left and right context tags, and similarly for words. $s(\cdot)$ denotes the score of the POS tag produced by the preprocessing tagger. The last row shows the “Character”-based features for Chinese. $\text{pre}_1(\cdot)$ and $\text{pre}_2(\cdot)$ denote the word prefixes with one and two characters respectively. $\text{suf}_1(\cdot)$ and $\text{suf}_2(\cdot)$ denote the word suffixes similarly. $c_n(\cdot)$ denotes the $n$-th character in the word. $\text{len}(\cdot)$ denotes the length of the word, capped at 5 if longer.

Table 2: Statistics of datasets.

Figure 4 shows the first- to third-order feature templates that we use in our model. We also use global features to capture the adjacent conjuncts agreement in a coordination structure, and the valency patterns for each POS category. Note that most dependency features are implicitly cross-task in that they include POS tag and segmentation information. For example, the standard feature involves the POS tags of the words on both ends of the arc.

5 Experimental Setup

5.1 Datasets

We evaluate our model on two Arabic datasets and one Chinese dataset. For the first Arabic dataset, we use the dataset used in the Statistical Parsing of Morally Rich Languages (SPMRL) Shared Task 2013 (Seddah et al., 2013). We follow the official split for training, development and testing set. We use the core set of 12 POS categories provided by Marton et al. (2013). In the second Arabic dataset, the training set is a dependency conversion of the Arabic Treebank, which primarily includes Modern Standard Arabic (MSA) text. However, we test on a new corpus, which consists of classical Arabic text obtained from the Comprehensive Islamic Library (CIS). A native Arabic speaker with background in computational linguistics annotated the morphological segmentation and POS tags. This corpus is an excellent testbed for a joint model because classical Arabic may use rather different vocabulary from MSA, while their syntactic grammars are very similar to each other. Therefore incorporating syntactic information should be particularly beneficial to morphological segmentation and POS tagging. For Chinese, we use the Chinese Penn Treebank 5.0 (CTB5) and follow the split in previous work (Zhang and Clark, 2010).

Table 2 summarizes the statistics of the datasets. For the SPMRL test set, we follow the common practice which limits the sentence lengths up to 70 (Seddah et al., 2013). For classical Arabic and Chinese, we evaluate on all the test sentences.

5.2 Generating Lattice Structures

In this section we introduce the methodology for constructing candidate sets for segmentation and POS tagging. Table 3 provides statistics on the generated candidate sets.

SPMRL 2013 Following Marton et al. (2013), we use the MADA system to generate candidate mor-
We use 10-fold cross validation to avoid overfitting on the training set.

Table 3: Quality of the lattice structures on each dataset. For SPMRL and CTB5, we show the statistics on the development sets. For classical Arabic, we directly show the statistics on the testing set because the development set is not available.

5.3 Evaluation Measures
Following standard practice in previous work (Hatori et al., 2012; Zhang et al., 2014a), we use F-score as the evaluation metric for segmentation, POS tagging and dependency parsing. We report the morpheme-level F-score for Arabic and the word-level F-score for Chinese. In addition, we use TedEval (Tsarfaty et al., 2012) to evaluate the joint prediction on the SPMRL dataset, because TedEval score is the only evaluation metric used in the official report. We directly use the evaluation tools provided on the SPMRL official website.7

5.4 Baselines
State-of-the-Art Systems For the SPMRL dataset, we directly compare with Björkelund et al. (2013). This system achieves the best TedEval score in the track of dependency parsing with predicted information and we directly republish the official result. We also compute the F-score of this system on each task using our own evaluation script.8 For the CTB5 dataset, we directly compare to the arc-eager system by Zhang et al. (2014a), which slightly outperforms the arc-standard system by Hatori et al. (2012).

System Variants We also compare against a pipeline variation of our model. In our pipeline model, we predict segmentations and POS tags by the same system that we use to generate candidates. The subsequent standard parsing model then operates on the predicted segmentations and POS tags.

---

7http://www.spmrl.org/spmrl2013-sharedtask.html
8F-score evaluation for Arabic is not straightforward due to the stem changes in the morphological analysis. Therefore, the comparison of F-scores is only approximate.
Table 4: Segmentation, POS tagging and unlabeled attachment dependency F-scores (%) and TedEval score (%) on different datasets. The first line denotes the performance by the pipeline variation of our model. The second row shows the results by our joint model. “Best Published” includes the best reported results: Björkelund et al. (2013) for the SPMRL 2013 shared task and Zhang et al. (2014a) for the CTB5 dataset. Note that the POS F-scores are not directly comparable because Björkelund et al. (2013) use a different POS tagset from us.

Figure 6: Absolute F-score (%) improvement of the joint model over the pipeline counterpart on seen and out-of-vocabulary (OOV) words.

5.5 Experimental Details

Following our earlier work (Zhang et al., 2014b), we train a first-order classifier to prune the dependency tree space. Following common practice, we average parameters over all iterations after training with passive-aggressive online learning algorithm (Crammer et al., 2006; Collins, 2002). We use the same adaptive random restart strategy as in our earlier work (Zhang et al., 2014b) and set $K = 300$. In addition, we also apply an aggressive early-stop strategy during training for efficiency. If we have found a violation against the ground truth during the first 50 iterations, we immediately stop and update the parameters based on the current violation. The reasoning behind this early-stop strategy is that weaker violations for some training sentences are already sufficient for separable training sets (Huang et al., 2012).

6 Results

Comparison to State-of-the-art Systems Table 4 summarizes the performance of our model and the best published results for the SPMRL and the CTB5 datasets. On both datasets, our system outperforms the baselines. On the SPMRL 2013 shared task, our approach yields a 2.1% TedEval score gain over the top performing system (Björkelund et al., 2013). We also improve the segmentation and dependency F-scores by 3.1% and 4.8% respectively. Note that the POS F-scores are not directly comparable because Björkelund et al. (2013) use a different POS tagset from us. On the CTB5 dataset, we outperform the state-of-the-art with respect to all tasks: segmentation (0.3%), tagging (0.1%), and dependency parsing (0.3%).

We are not aware of any published results on the Classical Arabic Dataset.

Zhang et al. (2014a) improve the dependency F-score to 82.14% by adding manually annotated intra-word dependency information. Even without such gold word structure annotations, our model still achieves a comparable result.
Impact of the Joint Prediction As Table 4 shows, our joint prediction model consistently outperforms the corresponding pipeline model in all three tasks. This observation is consistent with findings in previous work (Hatori et al., 2012; Tratz, 2013). We also observe that gains are higher (2%) on the classical Arabic dataset, which demonstrates that joint prediction is particularly helpful in bridging the gap between MSA and classical Arabic.

Figure 6 shows the break of the improvement based on seen and out-of-vocabulary (OOV) words. As expected, across all languages OOV words benefit more from the joint prediction, as they constitute a common source of error propagation in a pipeline model. The extent of improvement depends on the underlying accuracy of the preprocessing for segmentation and POS tagging on OOV words. For instance, we observe a higher gain (7%) on Chinese OOV words which have a 61.5% accuracy when processed by the original stand-alone POS tagger. On the SPMRL dataset, the gain on OOV words is lower (3%), while preprocessing accuracy is higher (82%). Their error reductions on OOV words are nevertheless close to each other. Table 5 summarizes the results on F-score error reduction.

We also observe that the error reductions of OOV words/morphemes on the Chinese and the Classical Arabic dataset are larger than that of the in-vocabulary counterparts (e.g. 26% vs. 20% on Chinese word segmentation). However, we have the opposite observation on the segmentation and POS tagging on the SPMRL dataset (28% vs. 48%). This can be explained by analyzing the oracle performance in which we select the best solution from possible candidates. The oracle error reduction of OOV morphemes in the SPMRL dataset is relatively low (44%), compared to the 61% oracle error reduction of OOV morphemes in the Classical Arabic dataset.

Impact of the Number of Alternative Analyses In Figure 7, we plot the performance on the SPMRL dataset as a function of the number $k$ of MADA analyses that we use to construct the candidate sets. For low $k$, increasing the number of analyses improves performance across all evaluation metrics. However, the performance converges at around $k = 15$.

Convergence Properties To assess the quality of the approximation obtained by the randomized greedy inference, we would like to compare it against the optimal solution. Following our earlier work (Zhang et al., 2014b), we use the highest score
among 3,000 restarts for each sentence as a proxy for the optimal solution. Figure 8 shows the normalized score of the retrieved solution as a function of the number of restarts. We observe that most sentences converge quickly. Specifically, more than 97% of the sentences converge within first 300 restarts. Since for the vast majority of cases our system converges fast, we achieve a comparable speed to that of other state-of-the-art joint systems. For example, our model achieves high performance on Chinese at about 0.5 sentences per second. The speed is about the same as that of the transition-based system (Hatori et al., 2012) with beam size 64, the setting that achieved best accuracy in their work.

Quality of Local Optima Figure 9 shows the cumulative distribution function (CDF) for the number of local optima versus the score of these local optima obtained from each restart. More specifically, the score captures the difference between a local optimum and the maximal score among 3,000 restarts. We can see that most of the local optima reached by hill-climbing have scores close to the maximum. For instance, about 30% of the local optima are identical to the best solution, namely $\text{score}_{\text{max}} - \text{score}_{\text{local}} = 0$.

7 Conclusions

In this paper, we propose a general randomized greedy algorithm for joint segmentation, POS tagging and dependency parsing. On both Arabic and Chinese, our model achieves improvement on the three tasks over state-of-the-art systems and pipeline variants of our system. In particular, we demonstrate that OOV words benefit more from the power of joint prediction. Finally, our experimental results show that increasing candidate sizes improves performance across all evaluation metrics.

Acknowledgments

This research is developed in a collaboration of MIT with the Arabic Language Technologies (ALT) group at Qatar Computing Research Institute (QCRI) within the Interactive Systems for Answer Search (IYAS) project. The authors acknowledge the support of the U.S. Army Research Office under grant number W911NF-10-1-0533, and of the DARPA BOLT program. We thank Meishan Zhang and Anders Björkelund for answering questions and sharing the outputs of their systems. We also thank the MIT NLP group and the ACL reviewers for their comments. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References


Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter, and Jan Hajic. 2013. Joint mor-


