Learning What’s Easy: Fully Differentiable Neural Easy-First Taggers

André F. T. Martins
Unbabel
& Instituto de Telecomunicações
Lisbon, Portugal
andre.martins@unbabel.com

Julia Kreutzer
∗Computational Linguistics
Heidelberg University, Germany
kreutzer@cl.uni-heidelberg.de

Abstract

We introduce a novel neural easy-first decoder that learns to solve sequence tagging tasks in a flexible order. In contrast to previous easy-first decoders, our models are end-to-end differentiable. The decoder iteratively updates a “sketch” of the predictions over the sequence. At its core is an attention mechanism that controls which parts of the input are strategically the best to process next. We present a new constrained softmax transformation that ensures the same cumulative attention to every word, and show how to efficiently evaluate and backpropagate over it. Our models compare favourably to BILSTM taggers on three sequence tagging tasks.

1 Introduction

In the last years, neural models have led to major advances in several structured NLP problems, including sequence tagging (Plank et al., 2016; Lample et al., 2016), sequence-to-sequence prediction (Sutskever et al., 2014), and sequence-to-tree (Dyer et al., 2015). Part of the success comes from clever architectures such as (bidirectional) long-short term memories (BILSTMs; Hochreiter and Schmidhuber (1997); Graves et al. (2005)) and attention mechanisms (Bahdanau et al., 2015), which are able to select the pieces of context relevant for prediction.

A noticeable aspect about many of the systems above is that they typically decode from left to right, greedily or with a narrow beam. While this is computationally convenient and reminiscent of the way humans process spoken language, the combination of unidirectional decoding and greediness leads to error propagation and suboptimal classification performance. This can partly be mitigated by globally normalized models (Andor et al., 2016) and imitation learning (Daumé et al., 2009; Ross et al., 2011; Bengio et al., 2015), however these techniques still have a left-to-right bias.

Easy-first decoders (Tsuruoka and Tsujii, 2005; Goldberg and Elhadad, 2010, §2) are an interesting alternative: instead of a fixed decoding order, these methods schedule their own actions by preferring “easier” decisions over more difficult ones. A disadvantage is that these models are harder to learn, due to the factorial number of orderings leading to correct predictions. Usually, gradients are not backpropagated over this combinatorial latent space (Kiperwasser and Goldberg, 2016), or a separate model is used to determine the easiest next move (Clark and Manning, 2016).

In this paper, we develop novel, fully differentiable, neural easy-first sequence taggers (§3). Instead of taking discrete actions, our decoders use an attention mechanism to decide (in a soft manner) which word to focus on for the next tagging decision. Our models are able to learn their own sense of “easiness”: the words receiving focus may not be the ones the model is most confident about, but the best to avoid error propagation in the long run. To make sure that all words receive the same cumulative attention, we further contribute with a new constrained softmax transformation (§4). This transformation extends the softmax by permitting upper bound constraints on the amount of probability a word can receive. We show how to evaluate this transformation and backpropagate its gradients.

We run experiments in three sequence tagging tasks: multilingual part-of-speech (POS) tagging, named entity recognition (NER), and word-level quality estimation (§5). We complement our findings with a visual analysis of the attention distribu-

∗This research was partially carried out during an internship at Unbabel.
Algorithm 1 Easy-First Sequence Tagging

Input: input sequence \( x_{1:L} \)
Output: tagged sequence \( \hat{y}_{1:L} \)

1: initialize \( B = S = \emptyset \)
2: while \( B \neq \{1, \ldots, L\} \) do
3: \( \text{for each non-covered position } i \notin B \) do
4: compute scores \( f(i, y_i; x_{1:L}, S) \), \( \forall y_i \)
5: end for
6: \( (j, \hat{y}_j) = \text{argmax}_{i, y_i} f(i, y_i; x_{1:L}, S) \)
7: \( S \leftarrow S \cup \{(j, \hat{y}_j)\} \), \( B \leftarrow B \cup \{j\} \)
8: end while

9: \( \hat{y}_{1:L} = \text{argmax}_{y_{1:L}} f(x_{1:L}, S, \hat{y}_{1:L}) \)
10: return \( \hat{y}_{1:L} \)

The high-level procedure is shown in Figure 1. We start by encoding the input sequence \( x_{1:L} \) as a matrix

\[
\text{Fig. 1: A neural easy-first system applied to a POS tagging problem. Given the current input/sketch representation, an attention mechanism decides where to focus (see bar plot) and is used to generate the next sketch. Right: A sequence of sketches \( (S^j)^N_{n=1} \) generated along the way.}
\]
where we denote by each word $i$ the feedforward attention proposed by Bahdanau et al. to focus on next. This is done in a similar way as the vanillal easy-first algorithm. However, it is possible, for example using convolutional layers. As stated above, our algorithm maintains a cumulative attention vector $\beta$, which progressively refine this sketch matrix, producing versions $S^1, \ldots, S^N$. The three operations above are repeated $N$ times (or “sketch steps”). The standard choice for $\rho$ is the softmax transformation. However, in this work, we consider other possible transformations (to be described in §4). After this, the cumulative attention is updated via $\beta = \beta^{n-1} + \alpha^n$.

### Sketch Generation.
Now that we have a distribution $\alpha^n \in \Delta^{L-1}$ over word positions, it remains to generate a sketch for those words. We first compute a single-state vector representation of the entire sentence $\bar{c}^n = \sum_{i=1}^{L} \alpha^n_i c_i$ as the weighted average of the word states defined in Eq. 1. Then, we update each column of the sketch matrix as:

$$s_i^n = s_i^{n-1} + \alpha^n_i \cdot \tanh(\text{affine}(c_i^n)), \forall i \in [L].$$

The intuition for this update is the following: in the extreme case where the attention distribution is peaked on a single word (say, the $k$th word, $\alpha^n_k = e_k$), we obtain $\bar{c}^n = c_k^n$ and the sketch update only affects that word, i.e.,

$$s_i^n = \begin{cases} s_i^{n-1} + \tanh(\text{affine}(c_i^n)) & \text{if } i = k \\ s_i^{n-1} & \text{if } i \neq k. \end{cases}$$

This is similar to the sketch update in the original easy-first algorithm (line 7 in Alg. 1).

The three operations above are repeated $N$ times (or “sketch steps”). The standard choice is $N = L$ (one step per word), which mimics the vanilla easy-first algorithm. However, it is possible to have fewer steps (or more, if we want the decoder to be able to “self-correct”). After completing the $N$ sketch steps, we obtain the final sketch matrix $S^N = [s_1^N, \ldots, s_L^N]$. Then, we compute a tag probability for every word as follows:

$$p_i = \text{softmax}(\text{affine}(\text{concat}(h_i, s_i^N))).$$

1. Note that this is a rank-one update, as it can be written in matrix notation as $S^N = S^{n-1} + \tanh(\text{affine}(\bar{c}^n)) \cdot \alpha^n$.  
2. We found the concatenation with $h_i$ in Eq. 6 beneficial to transfer input information directly to the output layer, avoiding the need of flowing this information through the sketches.

---

**Algorithm 2** Neural Easy-First Sequence Tagging

**Input:** input sequence $x_1:L$, sketch steps $N$  
**Output:** tagged sequence $y_1:L$  

```algorithm
1: initialize $\beta^0 = 0$ and $s_i^0 = 0$, $\forall i \in [L]$  
2: encode sequence as $[h_1, \ldots, h_L]$  
3: for $n = 1, 2, \ldots, N$ do  
4:  for each position $i \in [L]$ do  
5:  compute $c_i^n$ and $s_i^n$ (Eqs. 1–2)  
6:  end for  
7:  compute attention $\alpha^n = \rho(z^n; \beta^{n-1})$  
8:  for each position $i \in [L]$ do  
9:  refine sketch $s_i^n$ from $\alpha^n_i$ and $c_i^n$, via Eq. 4 (single-state) or Eq. 7 (full-state)  
10: end for  
11: $\beta^n = \beta^{n-1} + \alpha^n$  
12: end for  
13: for each position $i \in [L]$ do  
14:  $p_i = \text{softmax(affine(\text{concat}(h_i, s_i^N)))}$  
15: end for  
16: end for
```

$H = [h_1, \ldots, h_L] \in \mathbb{R}^{D_h \times L}$ (line 2). Our model is completely agnostic about this encoding step. In our experiments, we compute $H$ by composing a lookup embedding layer with a BiLSTM (Hochreiter and Schmidhuber, 1997; Graves et al., 2005), as this strategy was successful in similar structured prediction tasks. However, other choices are possible, for example using convolutional layers.

As stated above, our algorithm maintains a cumulative attention vector $\beta \in \mathbb{R}^L$ and a sketch matrix $S \in \mathbb{R}^{D_s \times L}$, both with all entries initialized to zero. It then performs $N$ sketching steps, which progressively refine this sketch matrix, producing versions $S^1, \ldots, S^N$. At the $n$th step, the following operations are performed:

**Input-Sketch Contextual Representation.** For each word $i \in [L]$, we compute a state $c_i^n$ summarizing the surrounding local information about other words and sketches. We use a simple vector concatenation over a $w$-wide context window:

$$c_i^n = \text{concat}(h_{i-w}, \ldots, h_{i+w}, s_{i-w}^{n-1}, \ldots, s_{i+w}^{n-1}),$$

where we denote by $s_j^n$ the $j$th column of $S^{n-1}$. The intuition is that the current sketches provide valuable information about the neighboring words’ predictions that can influence the prediction for the $i$th word. In the vanilla easy-first algorithm, this was assumed in the score computation (line 4 of Alg. 1).

**Attention Mechanism.** We then use an attention mechanism to decide what is the “best” word to focus on next. This is done in a similar way as the feedforward attention proposed by Bahdanau et al. (2015). We first compute a score for each word $i \in [L]$ based on its contextual representation,  

$$z_i^n = v^T \tanh(\text{affine}(c_i^n)),$$

where $v \in \mathbb{R}^{D_s}$ is a model parameter. Then, we aggregate these scores in a vector $z^n \in \mathbb{R}^L$ and apply a transformation $\rho$ to map them to a probability distribution $\alpha^n \in \Delta^{L-1}$ (optionally taking into account the past cumulative attention $\beta^{n-1}$):

$$\alpha^n = \rho(z^n; \beta^{n-1}).$$

The “standard” choice for $\rho$ is the softmax transformation. However, in this work, we consider other possible transformations (to be described in §4).

---

$(1)$
In §5, we compare this to a BILSTM tagger, which predicts according to \( p_i = \text{softmax}(\text{affine}(h_i)) \).

### 3.2 Full-State Model (NEF-F)

The full-state model differs from the single-state model in §3.1 by computing a full matrix for every sketch step, instead of a single vector. Namely, instead of Eq. 4, it does the following sketch update for every word \( i \in [L] \):

\[
s^n_i = s^{n-1}_i + \alpha^n_i \cdot \tanh(\text{affine}(c^n_i)). \tag{7}
\]

Note that the only difference is in replacing the single vector \( \bar{c}^n \) by the word-specific vector \( c^n_i \).

As a result, this is no longer a rank-one update of the sketch matrix, but a full update. In the extreme case where the attention is peaked on a single word, the sketch update reduces to the same form as in the single-state model (Eq. 5). However, the full-state model is generally more flexible and allows processing words in parallel, since it allows different sketch updates for multiple words receiving attention, instead of trying to force those words to receive the same update. We will see in the experiments (§5) that this flexibility can be important in practice.

### 3.3 Computational Complexity

For both models, assuming that the \( \rho(z; \beta) \) transformation in Eq. 3 takes \( O(L) \) time to compute, the total runtime of Alg. 2 is \( O((N + K)L) \) (where \( K \) is the number of tags), which becomes \( O(L^2) \) if \( K \leq N = L \). This is so because the input-sketch representation, the attention mechanism, and the sketch generation step all have \( O(L) \) complexity, and the final softmax layer requires \( O(KL) \) operations. This is the same runtime of the vanilla easy-first algorithm, though the latter can be reduced to \( O(KL \log L) \) with caching and a heap, if the scores in line 4 depend only on local sketches (Goldberg and Elhadad, 2010). By comparison, a standard BILSTM tagger has runtime \( O(KL) \).

### 4 Constrained Softmax Attention

An important part of our models is their attention component (line 7 in Alg. 2). To keep the “easy-first” intuition, we would like the transformation \( \rho \) in Eq. 3 to have a couple of properties:

1. **Sparsity**: being able to generate sparse distributions \( \alpha^n \) (ideally, peaked on a single word).
2. **Evenness**: over iterations, spreading attention evenly over the words. Ideally, the cumulative attention should satisfy \( \beta^n \in [0,1]^L \) for every \( n \in [N] \) and \( \beta^N = 1 \).

The standard choice for attention mechanisms is the softmax transformation, \( \alpha^n = \text{softmax}(z^n) \).

However, the softmax does not satisfy either of the properties above. For the first requirement, we could incorporate a “temperature” parameter in the softmax to push for more peaked distributions. However, this does not guarantee sparsity (only “hard attention” in the limit) and we found it numerically unstable when plugged in Alg. 2.

For the second one, we could add a penalty before the softmax transformation, \( \alpha^n = \text{softmax}(z^n - \lambda \beta^{n-1}) \), where \( \lambda \geq 0 \) is a tunable hyperparameter. This strategy was found effective to prevent a word to receive too much attention, but it made the model less accurate.

An alternative is the sparsemax transformation (Martins and Astudillo, 2016):

\[
\text{sparsemax}(z) = \arg \min_{\alpha \in \Delta^{L-1}} \| \alpha - z \|^2. \tag{8}
\]

The sparsemax maintains most of the appealing properties of the softmax (efficiency to evaluate and backpropagate), and it is able to generate truly sparse distributions. However, it still does not satisfy the “evenness” property.

Instead, we propose a novel constrained softmax transformation that satisfies both requirements. It resembles the standard softmax, but it allows imposing hard constraints on the maximal probability assigned to each word. Let us start by writing the (standard) softmax in the following variational form (Amari and Nagaoka, 2001):

\[
\text{softmax}(z) = \arg \min_{\alpha \in \Delta^{L-1}} \text{KL}(\alpha \| \text{softmax}(z)) = \arg \min_{\alpha \in \Delta^{L-1}} -\text{H}(\alpha) - z^\top \alpha, \tag{9}
\]

where \( \text{KL} \) and \( \text{H} \) denote the Kullback-Leibler divergence and the entropy, respectively. Based on this observation, we define the constrained softmax transformation as follows:

\[
\text{csoftmax}(z; u) = \arg \min_{\alpha \in \Delta^{L-1}} -\text{H}(\alpha) - z^\top \alpha \quad \text{s.t.} \quad \alpha \leq u, \tag{10}
\]

where \( u \in \mathbb{R}^L \) is a vector of upper bounds. Note that, if \( u \geq 1 \), all constraints are loose and this
reduces to the standard softmax; on the contrary, if \( u \in \Delta^{L-1} \), they are tight and we must have \( \alpha = u \) due to the normalization constraint. Thus, we propose the following for Eq. 3:

\[
\alpha^n = \text{csoftmax}(z^n; 1 - \beta^n - 1).
\]

The constraints guarantee \( \beta^n = \beta^n - 1 + \alpha^n \leq 1 \). Since \( \sum_{i=1}^N \alpha^n = N \), they also ensure that \( \beta^N = 1 \), hence the “evenness” property is fully satisfied. Intuitively, each word gets a credit of one unit of attention that is consumed during the execution of the algorithm. When this credit expires, all subsequent attention weights for that word will be zero.

The next proposition shows how to evaluate the constrained softmax and compute its gradients.

**Proposition 1** Let \( \alpha = \text{csoftmax}(z; u) \), and define the set \( A = \{ i \in [L] \mid \alpha_i < u_i \} \) of the constraints in Eq. 10 that are met strictly. Then:

- **Forward propagation.** The solution of Eq. 10 can be written in closed form as \( \alpha_i = \min\{\exp(z_i)/Z, u_i\} \), where \( Z = \sum_{i \in A} \exp(z_i) / \sum_{i \notin A} u_i \).

- **Gradient backpropagation.** Let \( L(\theta) \) be a loss function, \( d\alpha = \nabla_{\alpha} L(\theta) \) be the output gradient, and \( dz = \nabla_{z} L(\theta) \) and \( du = \nabla_{u} L(\theta) \) be the input gradients. Then, we have:

\[
\begin{align*}
  dz_i &= \mathbb{1}(i \in A) \alpha_i (d\alpha_i - m) \quad (12) \\
  du_i &= \mathbb{1}(i \notin A) (d\alpha_i - m) \quad (13)
\end{align*}
\]

where \( m = \left(\sum_{i \in A} \alpha_i \right) / \left(1 - \sum_{i \notin A} u_i\right) \).

**Proof:** See App. A (supplementary material).

Algs. 3–4 turn the results in Prop. 1 into concrete procedures for evaluating \text{csoftmax} and for backpropagating its gradient. Their runtimes are respectively \( O(L \log L) \) and \( O(L) \).

---

**Algorithm 3** Constrained Softmax Forward

**Input:** \( z, u \)  
**Output:** \( \alpha = \text{csoftmax}(z; u) \)

1: initialize \( s := 0, \mathcal{A} := [L], Z := \sum_{i=1}^K \exp(z_i) \)
2: sort \( q_i \geq ... \geq q_{i_k} \), where \( q_i = \exp(z_i)/u_i, \forall i \in [L] \)
3: for \( k = 1 \) to \( L \) do
4: \( \alpha_{i_k} := \exp(z_{i_k}) (1 - s)/Z \)
5: if \( \alpha_{i_k} > u_{i_k} \) then
6: \( Z := Z - \exp(z_{i_k}) \)
7: \( \alpha_{i_k} := u_{i_k}, \quad s := s + \alpha_{i_k}, \quad \mathcal{A} := \mathcal{A} \setminus \{i_k\} \)
8: end if
9: end for

**Algorithm 4** Constrained Softmax Backprop

**Input:** \( z, u, dx \) (and cached \( \alpha, \mathcal{A}, s \) from Alg. 3)  
**Output:** \( dz, du \)

1: \( m := \sum_{i \in \mathcal{A}} \alpha_i \) \( dx_i / (1 - s) \)
2: for \( i \in [L] \) do
3: if \( i \in \mathcal{A} \) then
4: \( dz_i := m u_i (d\alpha_i - m), \quad du_i := 0 \)
5: else
6: \( dz_i := 0, \quad du_i := d\alpha_i - m \)
7: end if
8: end for

---

5 Experiments

We evaluate our neural easy-first models in three sequence tagging tasks: POS tagging, NER, and word quality estimation.

5.1 Part-of-Speech Tagging

We ran POS tagging experiments in 12 languages from the Universal Dependencies project v1.4 (Nivre et al., 2016), using the standard splits. The datasets contain 17 universal tags.\(^3\)

We implemented Alg. 2 in DyNet (Neubig et al., 2017), which we extended with the constrained softmax operator (Algs. 3–4).\(^4\) We used 64-dimensional word embeddings, initialized with pre-trained Polyglot vectors (Al-Rfou et al., 2013). Apart from the words, we embedded prefix and suffix character \( n \)-grams with \( n \leq 4 \). We set the affix embedding size to 50 and summed all these embeddings; in the end, we obtained a 164-dimensional representation for each word (words, prefixes, and suffixes). We then fed these embeddings into a BILSTM (with 50 hidden units in each direction) to obtain the encoder states \([h_1, \ldots, h_L] \in \mathbb{R}^{100 \times L}\). The other hyperparameters were set as follows: we used a context size \( w = 2 \), set the pre-attention size \( D_z \) and the sketch size \( D_s \) to 50, and applied dropout with a probability of 0.2 after the embedding and BILSTM layers and before the final softmax output layer.\(^5\) We ran 20 epochs of Adagrad to minimize the cross-entropy loss, with a stepsize of 0.1, and gradient

---

\(^3\)Our choice of languages covers 8 families: Romance (French, Portuguese, Spanish), Germanic (English, German), Slavic (Czech, Russian), Semitic (Arabic), Indo-Iranian (Hindi), Austronesian (Indonesian), Sino-Tibetan (Chinese), and Japonic (Japanese). For all languages, we used the datasets that have the canonical language name, except for Russian, where we used the (much larger) SynTagRus corpus. We preferred large datasets over small ones, to reduce the impact of overfitting in our analysis.

\(^4\)The code is available at https://github.com/Unbabel/neural-easy-first.

\(^5\)These hyperparameters were tuned in the English development set, and kept fixed across languages.
Table 1: POS tagging accuracies. The average in the rightmost column is over the words of each treebank.

† Note that Gillick et al. (2016) and Plank et al. (2016) are not strictly comparable, since they use older versions of the treebanks (UD1.1 and UD1.2, respectively).

For comparison, we also show the accuracies reported by Gillick et al. (2016) for their byte-to-span system (trained separately on each language) and by Plank et al. (2016) for their state-of-the-art multi-task BILSTM tagger (these results are not fully comparable though, due to different treebank versions).

Among the neural easy-first systems, we observe that NEF-F with csoftmax attention generally outperforms the others, but the differences are very slight (excluding the sparsemax attention system, which performed substantially worse). This system wins over the linear system for all languages but Spanish, and over the BILSTM baseline for 9 out of 12 languages (loses in Arabic and German, and ties in Japanese). Note, however, that the differences are small (95.47% against 95.39%, averaged across treebanks). We conjecture that this is due to the fact that the BILSTM already captures most of the relevant context in its encoder. Our NEF-F system with csoftmax also wins over the vanilla easy-first system for 10 out of 12 languages (arguably due to its ability to backpropagate the gradients through the soft attention mechanism), but the difference in the average score is again small (95.47% against 95.42%).

Figure 2 depicts some patterns learned by the NEF-F model with various attention types. With the csoftmax, the model learns to move left and right, and the main verb “thought” is the most
prominent candidate for the easiest decision. In fact, in 57% of the test sentences, the model focuses first on a verb. The “raindrop” appearance of the plot is due to the evenness property of $csoftmax$, which causes the attention over a word to increase gradually until the cumulative attention is exhausted. This contrasts with the softmax attention (less diverse and non-sparse) and the sparsemax (sparse, but not even). We show for comparison the (hard) decisions made by the vanilla easy-first decoder.

### 5.2 Named Entity Recognition

Next, we applied our model to NER. We used the official datasets from the CoNLL 2002-3 shared tasks (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003), which tag names, locations, and organizations using a BIO scheme, and cover four languages (Dutch, English, German, and Spanish). We made two experiments: one using the exact same BILSTM and NEF models with a standard softmax output layer, as in §5.1 (which does not guarantee valid segmentations), and another one replacing the output softmax layer by a sequential CRF layer, which requires learning $O(K^2)$ additional parameters for pairs of consecutive tags (Huang et al., 2015; Lample et al., 2016). We used the same hyperparameters as in the POS tagging experiments, except the dropout probability, which was set to 0.3 (tuned on the validation set). For English, we used pre-trained 300-dimensional GloVe-840B embeddings (Pennington et al., 2014); for Spanish and German, we used the 64-dimensional word embeddings from Lample et al. (2016); for Dutch we used the aforementioned Polyglot vectors. All embeddings are fine-tuned during training. Since many words are not entities, and those receive a default “outside” tag, we expect that fewer sketch steps are necessary to achieve top performance.

Table 2 shows the results, which confirm this hypothesis. We compare the same BILSTM baseline to our NEF-S and NEF-F models with $csoftmax$ attention (with and without the CRF output layer), varying the maximum number of sketch steps. We also compare against the byte-to-span model of Gillick et al. (2016) and the state-of-the-art character-based LSTM-CRF system of Lample et al. (2016). We can see that, for all languages, the NEF-CRF-F model with 5 steps is consistently better than the BILSTM-CRF and, with the exception of English, the NEF-CRF-S model. The same holds for the BILSTM and NEF-F models without the CRF output layer. With the exception of German, increasing the number of steps did not make a big difference.

Figure 3 shows the attention distributions over the sketch steps for an English sentence, for full-state models trained with $N \in \{5, L\}$. The model with $L$ sketch steps learned that it is easiest to focus on the beginning of a named entity, and then to move to the right to identify the full span. The model with only 5 sketch steps learns to go straight to the point, placing most attention on the entity words and ignoring most of the o-tokens.

### 5.3 Word-Level Quality Estimation

Finally, we evaluate our model’s performance on word-level translation quality estimation. The goal is to evaluate a translation system’s quality without access to reference translations (Blatz et al., 2004; Specia et al., 2013). Given a sentence pair (a source sentence and its machine translated sentence in a target language), a word-level system classifies each target word as OK or BAD. We used the official English-German dataset from the WMT16 shared task (Bojar et al., 2016).

This task differs from the previous ones in which its input is a sentence pair and not a single sentence, mixing character and word-based models, sharing a model across languages, or combining CRFs with convolutional and recurrent layers. We used simpler models in our experiments since our goal is to assess how much the neural easy-first systems can bring in addition to a BILSTM system, rather than building a state-of-the-art system.

|               | Dut. | Eng. | Ger. | Spa. |
|---------------|------|------|------|------|
| Gillick et al. (2016) | 78.08 | 84.57 | 72.08 | 81.83 |
| Lample et al. (2016)   | **81.74** | **90.94** | **78.76** | **85.75** |

Table 2: $F_1$ scores for NER, computed by the CoNLL 2002 evaluation script.
sentence. We replaced the affix embeddings by
the concatenation of the 64-dimensional embed-
dings of the target words with those of the aligned
source words (we used the alignments provided in
the shared task), yielding 128-dimensional repre-
sentations. We used the same hyperparameters as
in the POS tagging task, except the dropout proba-
bility, set to 0.1. We followed prior work (Kreutzer
et al., 2015) and upweighted the BAD words in the
loss function to make the model more pessimistic;
we used a weight of 5 (tuned in the validation set).

Table 3 shows the results. We see that all our
NEF-S and NEF-F models outperform the BIL-
STM, and that the NEF-F model with 5 sketch
steps achieved the best results.7 Figure 4 illus-
trates the attention over the target words for 5
sketches. We observe that the attention focuses
early in the areas predicted BAD and moves left
and right within these areas, not wasting attention
on the OK part of the sentence. This block-wise fo-

7Our best system would rank third in the shared task, out
of 13 submissions. The winner system, which achieved 49.52
$F_1$-MULT, was considerably more complex than ours, using
an ensemble of three neural networks with a linear system
(Martins et al., 2016).

Table 3: $F_1$-MULT scores (product of $F_1$ for OK
and BAD words) for word-level quality estimation,
computed by the official shared task script.

|          | NEF-S | NEF-F |
|----------|-------|-------|
| N = 5    | N = L | N = 5 |
| BILSTM   | 39.71 | 40.84 |
| NEF-S    | 40.91 | 40.99 |
| NEF-F    | 41.18 |       |

Figure 4: Example for word-level quality estima-
tion. The source sentence is “To open the Actions
panel, from the main menu, choose Window > Ac-
tions.” BAD words are red (bold font), OK words
are green.

5.4 Ablation Study

To better understand our proposed model, we car-
rried out an ablation study for NER on the English
dataset. The following alternate configurations
were tried and compared against the NEF-CRF-F
model with csoftmax attention and 5 sketch steps:

- A NEF-CRF-F model for which the final con-
catenation in Eq. 6 was removed, being replaced
by $p_i = \text{softmax}(\text{affine}(s_i^N))$. The goal was
to see if the sketches retain enough information
about the input to make a final prediction with-
out requiring the states $h_i$.
- A model for which the attention mechanism ap-
plied at each sketch step was replaced by a uni-
form distribution over the input words.
- A vanilla easy-first system (Alg. 1). Since this
system can only focus on one word at the time
(unlike the models with soft attention), we tried
both $N = 5$ and $N = L$ sketch steps.
- A left-to-right and right-to-left model, which re-
places the attention mechanism by one of these
two prescribed orders.

Table 4 shows the results. As expected, the neural
easy-first system was the best performing one, al-
though the difference with respect to the ablated
Table 4: Ablation experiments. Reported are $F_1$ scores for NER in the English test set.

| System                                    | $F_1$ Score |
|-------------------------------------------|-------------|
| NEF-CRF-F, $N = 5$                        | 88.01       |
| NEF-CRF-F w/out concat, $N = 5$           | 87.47       |
| Uniform Attention, $N = 5$                | 87.46       |
| Vanilla EF + CRF, $N = 5$                 | 87.17       |
| Vanilla EF + CRF, $N = L$                 | 87.46       |
| Left-to-right + CRF, $N = L$              | 87.57       |
| Right-to-left + CRF, $N = L$              | 87.53       |

systems is relatively small. Removing the concatenation in Eq. 6 is harmful, which suggests that there is information about the input not retained in the sketches. The uniform attention performs surprisingly well, and so do the left-to-right and right-to-left models, but they are still about half a point behind. The vanilla easy-first system has the worst performance with $N = 5$. This is due to the fact that the vanilla model is incapable of processing words “in parallel” in the same sketch step, a disadvantage with respect to the neural easy-first models, which have this capability due to their soft attention mechanisms (see the top image in Fig. 3).

6 Related Work

Vanilla easy-first decoders have been used in POS tagging (Tsuruoka and Tsujii, 2005; Ma et al., 2013), dependency parsing (Goldberg and Elhadad, 2010), and coreference resolution (Stoyanov and Eisner, 2012), being related to cyclic dependency networks and guided learning (Toutanova et al., 2003; Shen et al., 2007). More recent works compute scores with a neural network (Socher et al., 2011; Clark and Manning, 2016; Kiperwasser and Goldberg, 2016), but they still operate in a discrete space to pick the easiest actions (the non-differentiable argmax in line 6 of Alg. 1). Generalizing this idea to “continuous” operations is at the very core of our paper, allowing gradients to be fully backpropagated. In a different context, building differentiable computation structures has also been addressed by Graves et al. (2014); Grefenstette et al. (2015).

An important contribution of our paper is the constrained softmax transformation. Others have proposed alternatives to softmax attention, including the sparsemax (Martins and Astudillo, 2016) and multi-focal attention (Globerson et al., 2016). The latter computes a KL projection onto a budget polytope to focus on multiple words. Our constrained softmax also corresponds to a KL projection, but (i) it involves box constraints instead of a budget, (ii) it is normalized to 1, and (iii) we also backpropagate the gradient over the constraint variables. It also achieves sparsity (see the “raindrop” plots in Figures 2–4), and is suitable for sequentially computing attention distributions when diversity is desired (e.g. soft 1-to-1 alignments). Recently, Chorowski and Jaitly (2016) developed an heuristic with a threshold on the total attention as a “coverage criterion” (see their Eq. 11), however their heuristic is non-differentiable.

Our sketch generation step is similar in spirit to the “deep recurrent attentive writer” (DRAW, Gregor et al. (2015)) which generates images by iteratively refining sketches with a recurrent neural network (RNN). However, our goal is very different: instead of generating images, we generate vectors that lead to a final sequence tagging prediction.

Finally, the visualization provided in Figures 2–4 brings up the question how to understand and rationalize predictions by neural network systems, addressed by Lei et al. (2016). Their model, however, uses a form of stochastic attention and it does not perform any iterative refinement like ours.

7 Conclusions

We introduced novel fully-differentiable easy-first taggers that learn to make predictions over sequences in an order that is adapted to the task at hand. The decoder iteratively updates a sketch of the predictions by interacting with an attention mechanism. To spread attention evenly through all words, we introduced a new constrained softmax transformation, along with an algorithm to back-propagate its gradients. Our neural-easy first decoder consistently outperformed a BILSTM on a range of sequence tagging tasks.

A natural direction for future work is to go beyond sequence tagging (which we regard as a simple first step) toward other NLP structured prediction problems, such as sequence-to-sequence prediction. This requires replacing the sketch matrix in Alg. 2 by a dynamic memory structure.

Acknowledgments

We thank the Unbabel research team and the reviewers for their comments. This work was supported by the Fundação para a Ciência e Tecnologia through contracts UID/EEA/50008/2013, PTDC/EEI-SII/7092/2014 (LearnBig), and CMUPERI/TIC/0046/2014 (GoLocal).
References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. arXiv preprint arXiv:1307.1662.

S. Amari and H. Nagaoka. 2001. Methods of Information Geometry. Oxford University Press.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In Proc. of the Annual Meeting of the Association for Computational Linguistics, pages 2442–2452.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In International Conference on Learning Representations.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In Advances in Neural Information Processing Systems, pages 1171–1179.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In Proc. of the International Conference on Computational Linguistics, page 315.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In Proceedings of the First Conference on Machine Translation, pages 131–198.

Jan Chorowski and Navdeep Jaitly. 2016. Towards better decoding and language model integration in sequence to sequence models. arXiv preprint arXiv:1612.02695.

Kevin Clark and Christopher D Manning. 2016. Improving coreference resolution by learning entity-level distributed representations. In Proc. of the Annual Meeting of the Association for Computational Linguistics.

H. Daumé, J. Langford, and D. Marcu. 2009. Search-based structured prediction. Machine learning, 75(3):297–325.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In Proc. of the Annual Meeting of the Association for Computational Linguistics, pages 334–343, Beijing, China. Association for Computational Linguistics.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In Proc. of the Annual Meeting of the North-American Chapter of the Association for Computational Linguistics.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In Proc. of the Annual Meeting of the Association for Computational Linguistics.

Y. Goldberg and M. Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In Proc. of Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 742–750.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In International Conference on Artificial Neural Networks, pages 799–804. Springer.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural Turing Machines. arXiv preprint arXiv:1410.5401.

Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. 2015. Learning to Transduce with Unbounded Memory. In Advances in Neural Information Processing Systems, pages 1819–1827.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. 2015. Draw: A recurrent neural network for image generation. In Proc. of the International Conference on Machine Learning, pages 1462–1471.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. arXiv preprint arXiv:1508.01991.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Easy-first dependency parsing with hierarchical tree lstms. arXiv preprint arXiv:1603.00375.

Julia Kreutzer, Shigehiko Schamoni, and Stefan Riezler. 2015. Quality estimation from scratch (quetch): Deep learning for word-level translation quality estimation. In Proceedings of the Tenth Workshop on Statistical Machine Translation, pages 316–322.
Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Je- Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Xuezhe Ma and Eduard Hovy. 2016. End-to-end se- Ji Ma, Tong Xiao, and Nan Yang. 2013. Easy-first pos andings. In Proc. of Empirical Methods for Natural Language Processing.

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

Xuezhe Ma and Eduard Hovy. 2016. End-to-end se-
quence labeling via bi-directional lstm-cnns-crf. In Proc. of the Annual Meeting of the Association for Computational Linguistics.

André F. T. Martins and Ramón Astudillo. 2016. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. In Proc. of the International Conference on Machine Learning.

André F. T. Martins and Ramón Astudillo. 2016. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. In Proc. of the International Conference on Machine Learning.

André F. T. Martins and Ramón Astudillo, Chris Hokamp, and Fábio Kepler. 2016. Unbabel’s participation in the wmt16 word-level translation quality estimation shared task. In Proceedings of the First Conference on Machine Translation.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. arXiv preprint arXiv:1701.03980.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Je- sus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Yevgeni Berzak, Riyaz Ahmad Bhat, Eckhard Bick, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Gülşen Cebirolu Eryiit, Giuseppe G. A. Celano, Fabrizio Chalub, Çar Çoltek, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Arantzazu Diaz de Irarrazá, Kaja Dobrovolec, Timothy Dozat, Kira Drogoanu, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richard Farkas, Jennifer Foster, Claudia Freitas, Katarina Gajošová, Daniel Galbraith, Marcos García, Moa Gärdenfors, Sebastian Garza, Filip Ginter, Lakes Goenaga, Koldo Gojenola, Memduh Gökmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matías Grioni, Nornmund Grúžitis, Bruno Guillaume, Jan Hajic, Linh Hà M, Dag Haug, Barbera Hladká, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hünner Kaşkara, Hiroshi Kanayama, Jenna Kanerva, Boris Kaiz, Jessica Kenney, Natalia Kotsyba, Si- mon Krek, Veronika Laippala, Lucia Lam, Phng Lê Hng, Alessandro Lenci, Nikoła Ljubesić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cátalina Máranda, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missili, Veriginca Mititelu, Yusuke Miyao, Simon- etta Montemagni, Keiko Sophie Mori, Shunsuke Morii, Bohdan Moskalevskyi, Kadri Muşcnekn, Mafia Mustafina, Kaili Mütüresep, Lng Nguyen Th, Huyn Nguyen Th Minh, Vitaly Nikolaev, Hanna Nurmi, Petya Osenova, Robert Östling, Lilja Øvrelid, Valeria Paiva, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkálna, Prokopis Prokopidis, Tiina Puolakainen, Sampio Pyysalo, Alexandre Rademaker, Loganathan Ra- masivy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Baiba Saulite, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Maria Šimková, Kiril Simov, Aaron Smith, Carolyn Spadine, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Lars Wallin, Jing Xian Wang, Jonathan North Washington, Mats Wirén, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanhí Zhu. 2016. Universal dependencies 1.4. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University in Prague.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation. Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2014), 12:1532–1543.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In Proc. of the Annual Meeting of the Association for Computational Linguistics (short papers).

Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and struc- tured prediction to no-regret online learning. In AIS- TATS, volume 1, page 6.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classifica- tion. In ACL, volume 7, pages 760–767.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011. Parsing natural scenes and natu- ral language with recursive neural networks. In Pro- ceedings of the 28th international conference on ma- chine learning (ICML-11), pages 129–136.
Lucia Specia, Kashif Shah, Jose G.C. de Souza, and Trevor Cohn. 2013. QuEst - a translation quality estimation framework. In Proc. of the Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 79–84.

Veselin Stoyanov and Jason Eisner. 2012. Easy-first coreference resolution. In COLING, pages 2519–2534.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems, pages 3104–3112.

E. F. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Proc. of International Conference on Natural Language Learning.

E.F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In Proc. of International Conference on Natural Language Learning.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1, pages 173–180. Association for Computational Linguistics.

Yoshimasa Tsuruoka and Jun’ichi Tsujii. 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. In Proceedings of the conference on human language technology and empirical methods in natural language processing, pages 467–474. Association for Computational Linguistics.
A Proof of Proposition 1

We provide here a detailed proof of Proposition 1.

A.1 Forward Propagation

The optimization problem is

\[ \text{csoftmax}(z, u) = \arg\min -H(\alpha) - z^\top \alpha \]
\[ \text{ s.t. } \begin{cases} 1^\top \alpha = 1 \\ 0 \leq \alpha \leq u. \end{cases} \]

The Lagrangian function is:

\[ L(\alpha, \lambda, \mu, \nu) = -H(\alpha) - z^\top \alpha + \lambda(1^\top \alpha - 1) - \mu^\top \alpha + \nu^\top (\alpha - u). \] (14)

To obtain the solution, we invoke the Karush-Kuhn-Tucker conditions. From the stationarity condition, we have

\[ 0 = \log(\alpha) + 1 - z + \lambda_1 - \mu + \nu, \] which due to the primal feasibility condition implies that the solution is of the form:

\[ \alpha = \exp(z + \mu - \nu)/Z, \] (15)

where \( Z \) is a normalization constant. From the complementarity slackness condition, we have that \( 0 < \alpha_i < u_i \) implies that \( \mu_i = \nu_i = 0 \) and therefore \( \alpha_i = \exp(z_i)/Z \). On the other hand, \( \nu_i > 0 \) implies \( \alpha_i = u_i \). Hence the solution can be written as \( \alpha_i = \min\{\exp(z_i)/Z, u_i\} \), where \( Z \) is determined such that the distribution normalizes:

\[ Z = \sum_{i \in A} \exp(z_i) \]
\[ 1 - \sum_{i \notin A} u_i, \] (16)

with \( A = \{i \in [L] \mid \alpha_i < u_i\} \).

A.2 Gradient Backpropagation

We now turn to the problem of backpropagating the gradients through the constrained softmax transformation. For that, we need to compute its Jacobian matrix, i.e., the derivatives \( \frac{\partial \alpha_i}{\partial z_j} \) and \( \frac{\partial \alpha_i}{\partial u_j} \) for \( i, j \in [L] \).

Let us first express \( \alpha \) as

\[ \alpha_i = \begin{cases} \frac{\exp(z_i)(1-s)}{\sum_{j \in A} \exp(z_j)}, & i \in A \\ \frac{u_i}{s}, & i \notin A, \end{cases} \] (17)

where \( s = \sum_{j \notin A} u_j \). Note that we have \( \partial s/\partial z_j = 0, \forall j \), and \( \partial s/\partial u_j = 1 \) \( (j \notin A) \). To compute the entries of the Jacobian matrix, we need to consider several cases.

**Case 1:** \( i \in A \). In this case, the evaluation of Eq. 17 goes through the first branch. Let us first compute the derivative with respect to \( u_j \). Two things can happen: if \( j \in A \), then \( s \) does not depend on \( u_j \), hence \( \partial s/\partial u_j = 0 \). Else, if \( j \notin A \), we have

\[ \frac{\partial \alpha_i}{\partial u_j} = -\frac{\exp(z_i) \partial s}{\sum_{k \in A} \exp(z_k)} = -\alpha_i/(1-s). \]

Now let us compute the derivative with respect to \( z_j \). Three things can happen: if \( j \in A \) and \( i \neq j \), we have

\[ \frac{\partial \alpha_i}{\partial z_j} = -\frac{\exp(z_i) \exp(z_j)(1-s)}{(\sum_{k \in A} \exp(z_k))^2} = -\alpha_i \alpha_j/(1-s). \] (18)
If $j \in A$ and $i = j$, we have

$$\frac{\partial \alpha_i}{\partial z_i} = (1 - s) \times \frac{\exp(z_i) \sum_{k \in A} \exp(z_k) - \exp(z_i)^2}{(\sum_{k \in A} \exp(z_k))^2} = \alpha_i - \alpha_j^2/(1 - s).$$  (19)

Finally, if $j \notin A$, we have $\frac{\partial \alpha_i}{\partial z_j} = 0$.

**Case 2:** $i \notin A$. In this case, the evaluation of Eq. 17 goes through the second branch, which means that $\frac{\partial \alpha_i}{\partial z_j} = 0$, always. Let us now compute the derivative with respect to $u_j$. This derivative is always zero unless $i = j$, in which case $\frac{\partial \alpha_i}{\partial u_j} = 1$.

To sum up, we have:

$$\frac{\partial \alpha_i}{\partial z_j} = \begin{cases} 1 & \text{if } i = j \text{ and } i \in A \\ 0, & \text{otherwise} \end{cases}$$  (20)

and

$$\frac{\partial \alpha_i}{\partial u_j} = \begin{cases} -\frac{\alpha_i}{1 - s}, & \text{if } i \in A, j \notin A \\ 1, & \text{if } i, j \notin A, i = j \\ 0, & \text{otherwise} \end{cases}$$  (21)

Therefore, we obtain:

$$dz_j = \sum_i \frac{\partial \alpha_i}{\partial z_j} d\alpha_i = 1(j \in A) \left( \alpha_j d\alpha_j - \frac{\alpha_j \sum_{i \in A} \alpha_i d\alpha_i}{1 - s} \right) = 1(j \in A) \alpha_j (d\alpha_j - m),$$  (22)

and

$$du_j = \sum_i \frac{\partial \alpha_i}{\partial u_j} d\alpha_i = 1(j \notin A) \left( d\alpha_j - \frac{\sum_{i \in A} \alpha_i d\alpha_i}{1 - s} \right) = 1(j \notin A) (d\alpha_j - m),$$  (23)

where $m = \frac{\sum_{i \in A} \alpha_i d\alpha_i}{1 - s}$. 