Replacing Greedy Search with Beam Search in Syntactic Parsing

Error States and Early Updates for Beam Search Training and Inference in a Syntactic Parser

Hannes Bengtsson & Patrik Habbe

Group 16

2024-03-13

LINKÖPINGS UNIVERSITET

Agenda

1. Project Scope and Method
2. Experiment Results and Conclusions
3. Project Motivation and Scientific Resources

Agenda

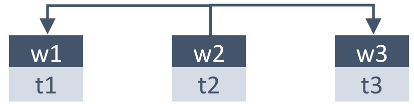
1. Project Scope and Method
2. Experiment Results and Conclusions
3. Project Motivation and Scientific Resources

We have replaced the greedy search algorithm with beam search for syntactic parsing and found modest improvements

Baseline System

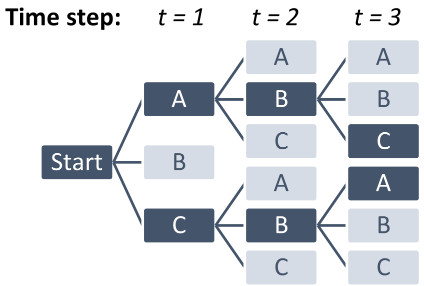
Syntactic Parsing

- Creating a formal representation of a sentence structure (in our use case dependency trees)
- We use a simple model that consider a fixed number of words and its corresponding POS tags to find “good” dependency trees



Greedy Search vs Beam Search

- Greedy search take best local score, beam search aim to find the best global score



Beam width of two, keep top two scoring candidates. Greedy only consider best at each time step

Extensions

“In this project we have replaced the greedy search for parsing in the baseline with a beam search and trained the fixed-window model for global scoring using two different methods.”

Method 1: Using Early Update

Inspired by *” Globally Normalized Transition-Based Neural Networks”* by Andor et al. (2016)

Method 2: Introducing Error States

Inspired by *” Efficient Structured Inference for Transition-Based Parsing with Neural Networks and Error States”* by Vaswani and Sagae (2016)

Datasets and Evaluation

Datasets

- Two dataset from the Universal Dependencies treebanks
- One English (EN) and one Italian (IT)
- Both needed to be projectivized

Accuracy

- The percentage of correctly predicted part-of-speech (POS) tags
- 88% for EN, 93% for IT

Unlabeled Attachment Score (UAS)

- The percentage or correctly predicted head positions using predicted POS tags from the tagger

Training beam search with early updates, and global normalization displayed improvements with wider beams, compared to error state training

1: Early updates and Loss function

Loss function:

$$L_{global-beam}(d^*_{1:j}\,;\,\theta) = -\sum_{i=1}^{j} \rho(\,d^*_{1:i-1}, d^*_i;\,\theta) + \ln \sum_{d'_{1:j} \in \mathcal{B}_j} exp \sum_{i=1}^{j} \rho(d'_{1:i-1}, d'_i;\,\theta)$$

The essentials:

d^* - The gold path

$\mathcal{B}_j$ - All paths in the beam at step j, with the gold path d^*

- If the gold beam falls out of the beam at step j, we perform an SGD step with the loss function above.
- If the gold beam stays until the end of decoding, $\mathcal{B}_j$ is replaced with $\mathcal{B}_n$, the beam without appending the gold path.

2: Error states for training and inference

Idea:

Introduce error states during the local training processes to account for incorrect derivation paths, normally not considered by locally trained models

Implementation:

- Revised the code in the oracle to also output error states if some moves are valid but not gold path.
- This allows the model to train not only on correct actions, but also on error states.
- Makes it possible to achieve global scoring using locally trained models.

For example, if the gold move is to shift from the buffer to our stack, we introduce the possibility of Left-Arc (LA) and Right-Arc (RA), if they are valid moves, that lead to an error state (ER).

Limitations

Summary of limitations:

We implemented beam search without adjusting the default hyperparameters or altering the underlying neural network's structure.

Details:

- Limited time prevented hyperparameter optimization
    - Default hyperparameters applied
- Baseline code's model features utilized as-is
- Model complexity unmodified (e.g., default hidden layer size)
- Beam search applied only during inference in the tagger

Agenda

1. Project Scope and Method
2. Experiment Results and Conclusions
3. Project Motivation and Scientific Resources

LINKÖPINGS UNIVERSITET

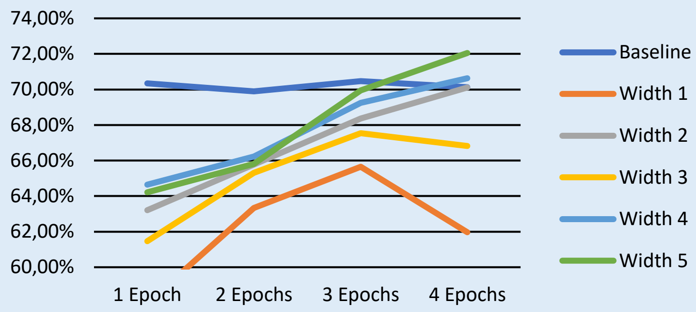
The early update implementation outperform both the baseline and error states implementation when using more epochs

Beam Search Experiment Results

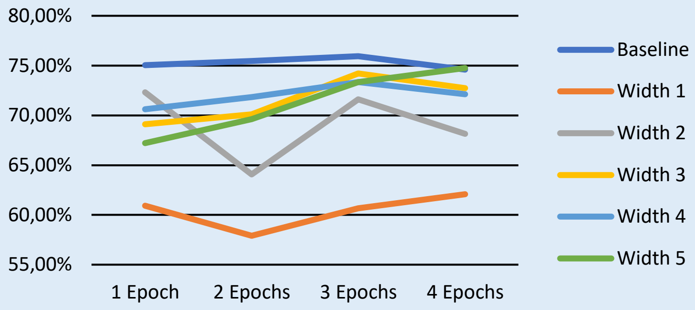
Early Updates

1

UAS[1] for different beam widths on the English dataset



UAS for different beam widths on the Italian dataset



General findings

- Wider beam width increases. inference performance in conjunction with an increase in the number of epochs
- Lower UAS than research paper (dataset, hyperparameters, etc.)
- On average 10 % of errors occurs “due to” beam search

English dataset

- Width 4 & 5 outperform baseline after 4 epochs

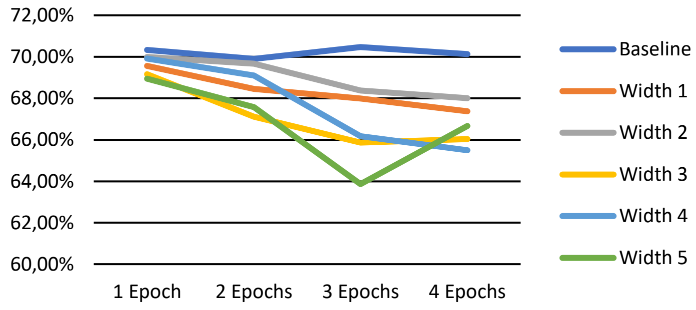
Italian dataset

- Beam width 5 for 4 epochs outperform baseline after 4 epochs with a UAS of 74,77%.

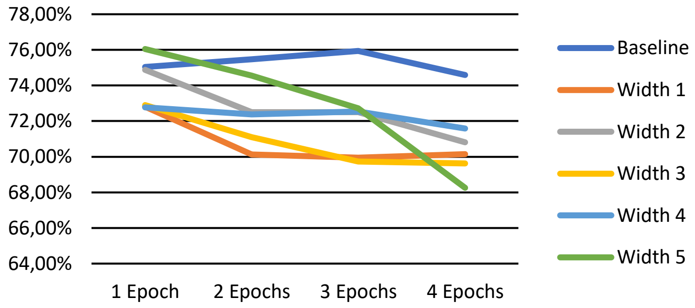
Error States

2

UAS[1] for different beam widths on the English dataset



UAS for different beam widths on the Italian dataset



General findings

- Increasing epoch degrades performance (overfitting)
- Lower UAS than article (dataset, hyperparameters, etc.)
- On average 5 % of errors occurs “due to” beam search

English dataset

- No beam outperforms the baseline

Italian dataset

- Beam width 5 for 1 epoch outperforms baseline with an UAS of 76.04%
- Best baseline, UAS of 75.94% on 3 epochs

1) Calculated UAS scores are when using gold label tags, it was more a concern about runtime than an active choice.

Our conclusions are that a more complex search algorithm may lead to limited increase in performance if the underlying model is very simple

Project Conclusions

Key project findings	Possible explanation
More epochs improved early update UAS score, but it deteriorates for error states	This make sense because one epoch in the early update contains fewer examples than one epoch for error states
The performance of both implementations are lower than that of Andor et al. (2016) as well as for Vaswani and Sagae (2016)	Lack of hyperparameter fine-tuning (highlighted as important in both paper), different datasets, simpler architecture for the underlying model
Early update achieves a higher UAS score than error states	Trying to comparing the results of Andor et al. (2016) and Vaswani and Sagae (2016) this seems to be the case as well. In the paper error states don't improve performance for beams larger than 4 without pre-trained embeddings
Both implementation seems to be doing what they are supposed to	Only beam search during inference and not training (or vice versa) deteriorates UAS score, the optimization verification test indicates that beam search works
Beam search may and may not lead to improved performance for syntactic parsing	A complex search algorithm may lead to limited increase in performance if the underlying model is too simple or the chosen hyperparameters are flawed

Agenda

1. Project Scope and Method
2. Experiment Results and Conclusions
3. Project Motivation and Scientific Resources

LINKÖPINGS UNIVERSITET

We chose beam search because we are currently writing our masters' thesis and consider beam search as a possible approach to obtain better results

Why we have chosen to implement beam search

- Beam search can be used on various NLP tasks, and it is a technique still used by e.g., OpenAI
- Implementing beam search seemed like a fair challenge (however it was much harder than expected)
- We found the idea behind beam search, to consider alternative "solutions", appealing as it made sense intuitively
- We are currently writing our masters' thesis on natural language processing and found beam search interesting as it might be applicable in out thesis work

Sources of Scientific Information

- *"Globally Normalized Transition-Based Neural Networks"*, Andor et al. (2016)
- *"Efficient Structured Inference for Transition-Based Parsing with Neural Networks and Error States"*, Vaswani and Sagae (2016)
- *"Structured Training for Neural Network Transition-Based Parsing"*, Weiss et al. (2015)
- *"A Fast and Accurate Dependency Parser using Neural Networks"*, Chen and Manning (2014)
- DeepLearning.AI and Andrew Ng on Sequence-to-Sequence Models, more specifically error analysis on beam search and "the optimization verification test"