TDDE09

Project G08

Exploring the performance of multilingual tagger-parser implementations

Contents

Method

What did we do in this project?

We wanted to **extend** the **baseline** project with improvements in order to increase the performance.

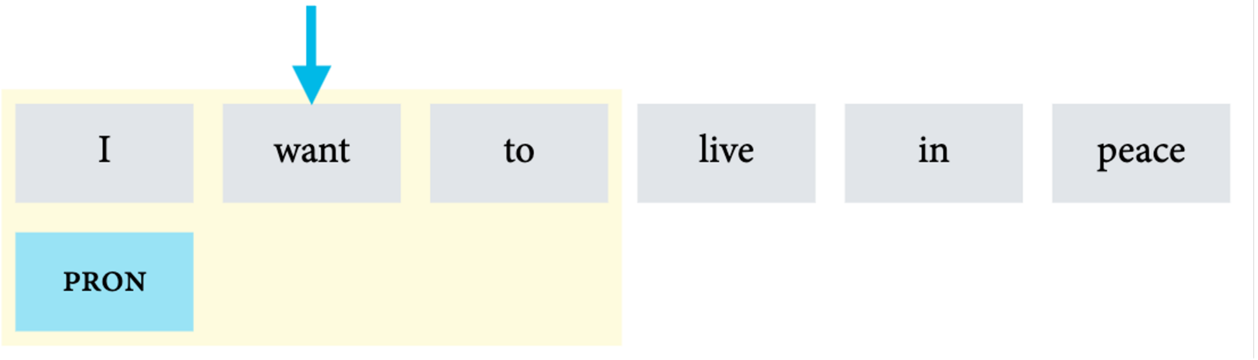
What was the baseline project?

The baseline project was a **tagger-parser** pipeline. The tagger was a simple auto-regressive tagger. The syntactic **transition-based** parser used the arc-standard system and was trained using a **static oracle**.

How was it improved upon?

We implemented a **dynamic oracle**, which aims to improve the training phase by exploring non-optimal paths. We also needed to implement the arc-hybrid system, in order to utilize the dynamic oracle, since it is arc-decomposable.

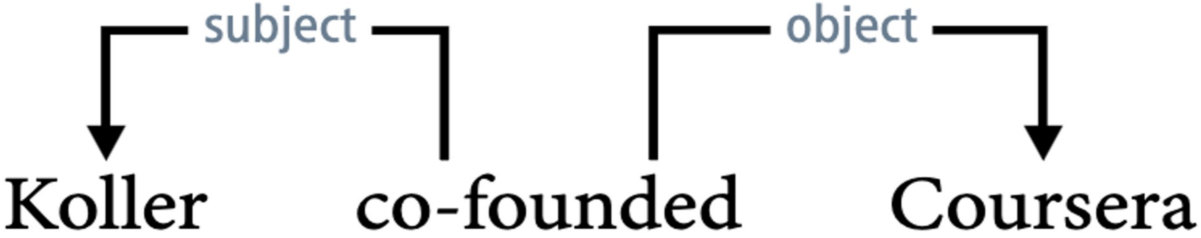
What is an autoregressive tagger?



The autoregressive tagger aims to **predict the tag** (Pronoun, Direct Object, etc.) for each word, using already predicted tags as well as other words in the window (marked as yellow).

https://www.ida.liu.se/~TDDE09/lectures/unit4/nlp-2024-402.pdf

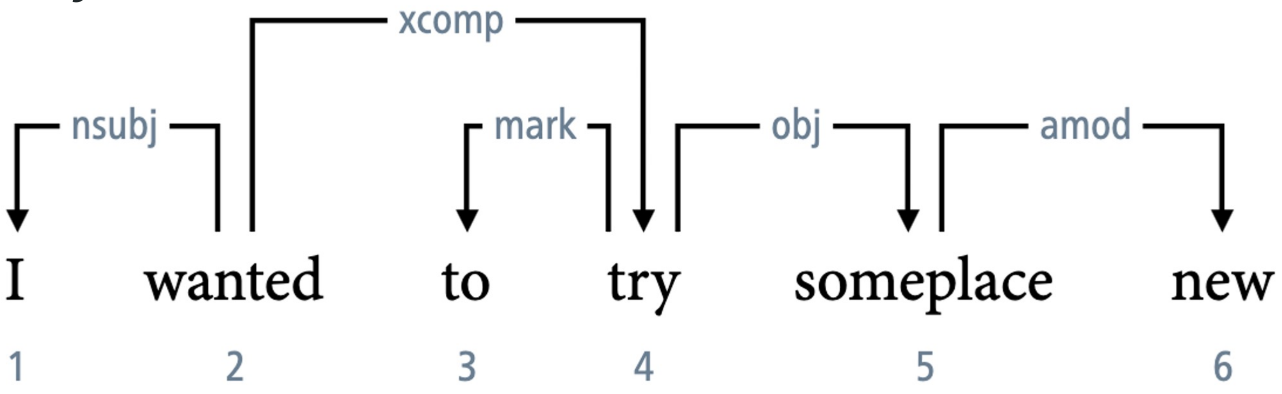
What is a transition-based parser?



A parser analyzes the syntactic structure of a sentence by identifying the relationships between words using tags. It focuses on understanding how words relate to each other within the sentence, forming a **hierarchical structure** that represents the sentence's syntactic organization.

https://www.ida.liu.se/~TDDE09/lectures/unit4/nlp-2024-404.pdf

Dependency tree illustration:

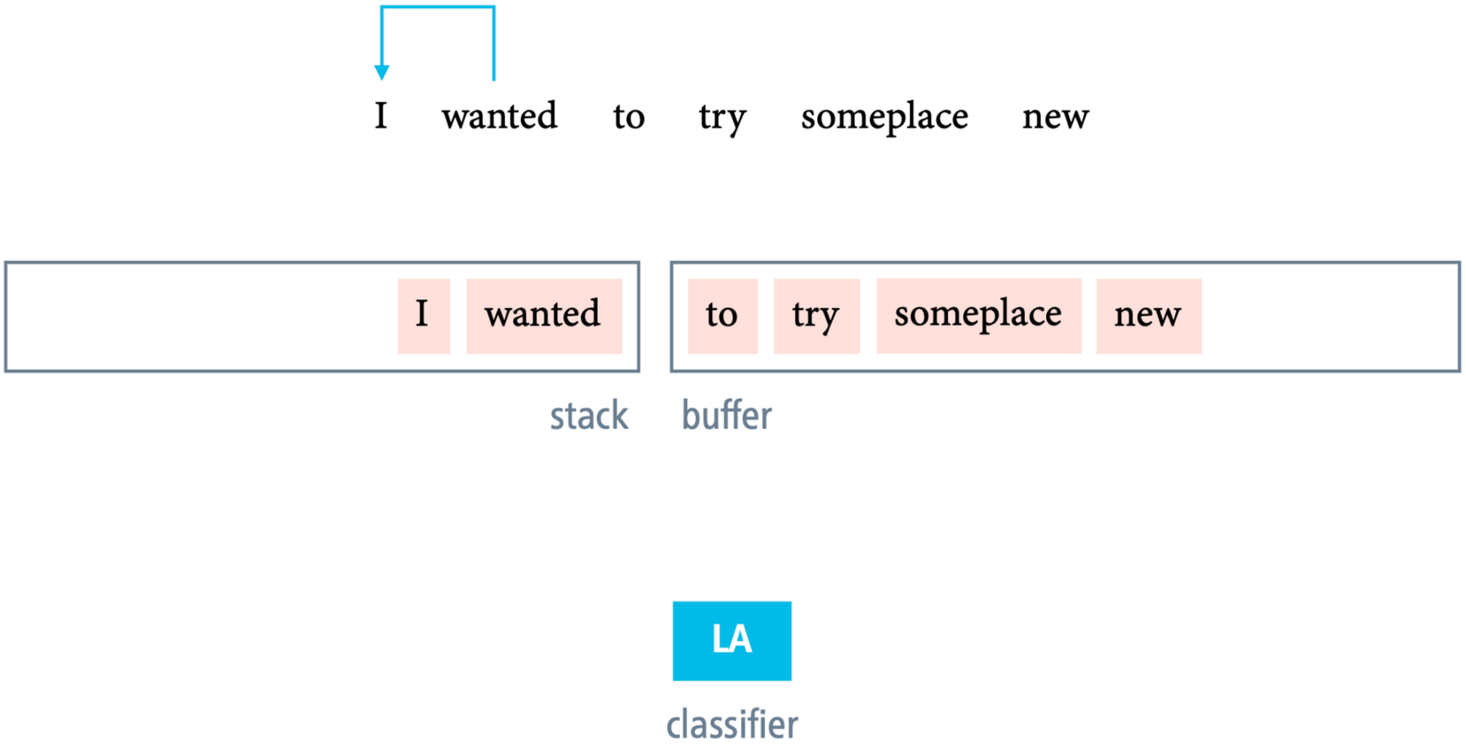


Dependency tree as an array:

word position	1	2	3	4	5	6
head position	2	0	4	2	4	5

https://www.ida.liu.se/~TDDE09/lectures/unit4/nlp-2024-404.pdf

Deciding the transition:



https://www.ida.liu.se/~TDDE09/lectures/unit4/nlp-2024-404.pdf

What is an oracle?

An oracle takes the role of a teacher during the training of the parser model. It is an algorithm that **takes** a gold-standard **dependency tree** and **generates** the gold-standard **transition sequence**. These are then used as training data for the parser model.

What is the difference between static and dynamic oracles?

A static oracle assumes that there is **one correct sequence of transitions** to take. If the model deviates from the oracle's path, it is forced to back on (teacher-forcing). Due to this a static oracle's predicted transitions can be pre-generated.

A dynamic oracle gives us valid zero-cost moves* **during the training** of the parser. This allows for the **choosing** of different, sometimes sub-optimal, paths for training, which is used to make the model more resistant to error-propagation.
It can therefore not be pre-generated.

Exploration

Exploration allows the dynamic oracle to try out **non-optimal moves** to potentially let it discover better strategies, improving performance.

Exploration parameters:

- Threshold for initiating exploration, denoted as **k**.
- Probability threshold that dictates the occurrence of exploration, denoted as **p**.

Ignore configurations where no valid moves are found.

Algorithm 3 Online training with a dynamic oracle

$\mathbf{w} \leftarrow 0$
for $I = 1 \rightarrow$ ITERATIONS **do**
    for sentence x with gold tree G_{gold} in corpus **do**
        $c \leftarrow c_s(x)$
        while c is not terminal **do**
            $t_p \leftarrow \arg\max_t \mathbf{w} \cdot \phi(c,t)$
            ZERO_COST $\leftarrow \{t \mid o(t; c, G_{\text{gold}}) = \textbf{true}\}$
            $t_o \leftarrow \arg\max_{t \in \text{ZERO\_COST}} \mathbf{w} \cdot \phi(c,t)$
            if $t_p \notin$ ZERO_COST **then**
                $\mathbf{w} \leftarrow \mathbf{w} + \phi(c, t_o) - \phi(c, t_p)$
            $t_n \leftarrow$ CHOOSE_NEXT(I,t_p,ZERO_COST)
            $c \leftarrow t_n(c)$
return $\mathbf{w}$

function CHOOSE_NEXT$_{\text{AMB}}$(I,t,ZERO_COST)
    if $t \in$ ZERO_COST **then**
        return t
    else
        return RANDOM_ELEMENT(ZERO_COST)

function CHOOSE_NEXT$_{\text{EXP}}$(I,t,ZERO_COST)
    if $I > k$ and RAND() $> p$ **then**
        return t
    else
        return CHOOSE_NEXT$_{\text{AMB}}$(I,t,ZERO_COST)

A Dynamic Oracle for Arc-Eager Dependency Parsing, Goldberg & Nivre, COLING 2012

Results

Results

All results are averages over 5 differently seeded runs.

We choose our batch size and our k value from *A Dynamic Oracle for Arc-Eager Dependency Parsing* by Goldberg & Nivre.

We got the best improvement from **p = 0.1** and choose it for our other runs.

p = 1.0 corresponds to **no exploration/ambiguity** and it gave the worst result.

parameters		k = 2	batch = 15		
system/probability p =	1.0	0.9	0.5	0.1	0.0
hybrid:dynamic	73.28	73.95	73.94	74.00	73.63

Table 1: Unlabeled Attachment Score (UAS) for english with golden tags

parameters	p = 0.1	k = 2	batch = 15
system/language	english	japanese	swedish
standard:static	73.58	85.67	71.00
hybrid:static	74.54	86.75	69.73
hybrid:dynamic	74.00	79.07	68.82

Table 2: Unlabeled Attachment Score (UAS) for languages using gold tags

parameters	p = 0.1	k = 2	batch = 15
system/language	english	japanese	swedish
standard:static	68.71	83.99	63.53
hybrid:static	69.50	85.06	62.42
hybrid:dynamic	68.70	77.50	61.69

Table 3: Unlabeled Attachment Score (UAS) for languages using generated tags

Discussion

Exploration vs Ambiguity (Goldberg & Nivre 2012)

parameters		k = 2	batch = 15		
system/probability p =	1.0	0.9	0.5	0.1	0.0
hybrid:dynamic	73.28	73.95	73.94	74.00	73.63

Table 1: Unlabeled Attachment Score (UAS) for english with golden tags

	ARA	BAS	CAT	CHI	CZE	ENG	GRE	HUN	ITA	TUR
	Unlabeled Attachment Scores									
Static	80.60	74.10	91.21	84.13	78.00	86.24	79.16	77.75	84.11	79.02
Dynamic-ambiguity	80.72	74.90	91.09	83.62	78.38	86.83	79.48	76.17	84.52	78.97
Dynamic-explore	83.06	76.10	92.01	84.65	79.54	88.81	80.66	77.10	84.77	78.84

Our results suggest a **0.72% increase** when using **exploration**, as opposed to ambiguity. This is **consistent** with, but not quite as large as the 1.98% increase, as seen in the literature.

A Dynamic Oracle for Arc-Eager Dependency Parsing, Goldberg & Nivre, COLING 2012

Comparison with literature (Goldberg & Nivre 2013)

parameters	p = 0.1	k = 2	batch = 15
system/language	english	japanese	swedish
standard:static	73.58	85.67	71.00
hybrid:static	74.54	86.75	69.73
hybrid:dynamic	74.00	79.07	68.82

Table 2: Unlabeled Attachment Score (UAS) for languages using gold tags

system / language	hungarian	chinese	greek	czech	basque	catalan	english	turkish	arabic	italian
					UAS					
hybrid:static	76.39	84.96	79.40	79.71	73.18	91.30	86.43	75.91	83.43	83.43
hybrid:dynamic	77.54	85.10	80.49	80.07	73.70	91.06	87.62	76.90	84.04	83.83

Training Deterministic Parsers with Non-Deterministic Oracles, Goldberg & Nivre, TACL 2013

Comparison with literature

system/language	english
standard:static	73.58
hybrid:static	74.54
hybrid:dynamic	74.00

english
86.43
87.62

Our results suggest a **0.54% decrease** when comparing the dynamic oracle to the static oracle with hybrid, the literature, however, suggests a **1.19% increase**.

A potential explanation is that our implementation is lacking somewhere. We tried to stick to *Algorithm 3,* however the case where there are **no valid moves** is unclear and might differ.

Another point of note is that we are using different treebanks for the english: *English Web Treebank LDC2012T13* vs. *CoNLL 2007 data set*. We could not find the exact math for the english data set from the article.

Training Deterministic Parsers with Non-Deterministic Oracles, Goldberg & Nivre, TACL 2013

Conclusions

Conclusions

- For parsing, the **arc-hybrid** system trained with a **static oracle** performs the **best on 2 out of 3 languages tested**. Only Swedish had higher accuracy using the arc-standard system with static oracle

- The literature suggests that the **dynamic oracle using the arc-hybrid system should perform best**, however we were unable to reproduce this

- Further testing with **different parsing systems** (arc-eager etc.) is required, and also maybe tuning the hyper-parameters for exploration, like threshold **k**, and **batch-size**

Questions?