Lemmatization

TA session

What is lemmatization?

Lemmatization is the process of grouping together different inflected forms of the same word. It's used in computational linguistics, natural language processing (NLP) and chatbots. Lemmatization links similar meaning words as one word, making tools such as chatbots and search engine queries more effective and accurate.

The goal of lemmatization is to reduce a word to its root form, also called a lemma. For example, the verb "running" would be identified as "run." Lemmatization studies the morphological, or structural, and contextual analysis of words.

How does lemmatization work?

Lemmatization takes a word and breaks it down to its lemma. For example, the verb "walk" might appear as "walking," "walks" or "walked." Inflectional endings such as "s," "ed" and "ing" are removed. Lemmatization groups these words as its lemma, "walk."

A basic way to perform lemmatization is to use an algorithm based on dictionary lookups. This process requires a detailed dictionary so the algorithm can find a specific word and link it back to the word's lemma. More complicated word forms or languages can require a rule-based system for lemmatization.

Applications of lemmatization

Lemmatization is commonly applied in the following areas:

- Artificial intelligence (AI).
- Big data analytics.
- Chatbots.
- Machine learning (ML).
- NLP.
- Search queries.
- Sentiment analysis.

Lemmatization vs. stemming

In linguistics, lemmatization is closely related to stemming, as both strip prefixes and suffixes that have been added to a word's base form.

Stemming algorithms cut off the beginning or end of a word using a list of common prefixes and suffixes that might be part of an inflected word. This process is generally indiscriminate and can result in base forms of a word with incorrect spelling or meaning. Stemming operates without any contextual knowledge, meaning that it can't discern between similar words with different meanings.

For example, the stem of "studies" and "studying" would be "studi" and "study," while in lemmatization the base form would be "study" for both "studies" and "studying." But both lemmatization and stemming would still have the same base form for the word "walking," for example. While being less accurate, stemming is easier to implement and runs faster. An example of stemming and lemmatization is shown as follows:

Stemming:	**Lemmatization:**
Study → Studi	**Study → Study**
Studying → Studi	**Studying → Study**
Studies → Studi	**Studies → Study**
Studied → Studi	**Studied → Study**
Studier → Studier	**Studier → Study**

Lemmatization advantages and disadvantages

Advantages

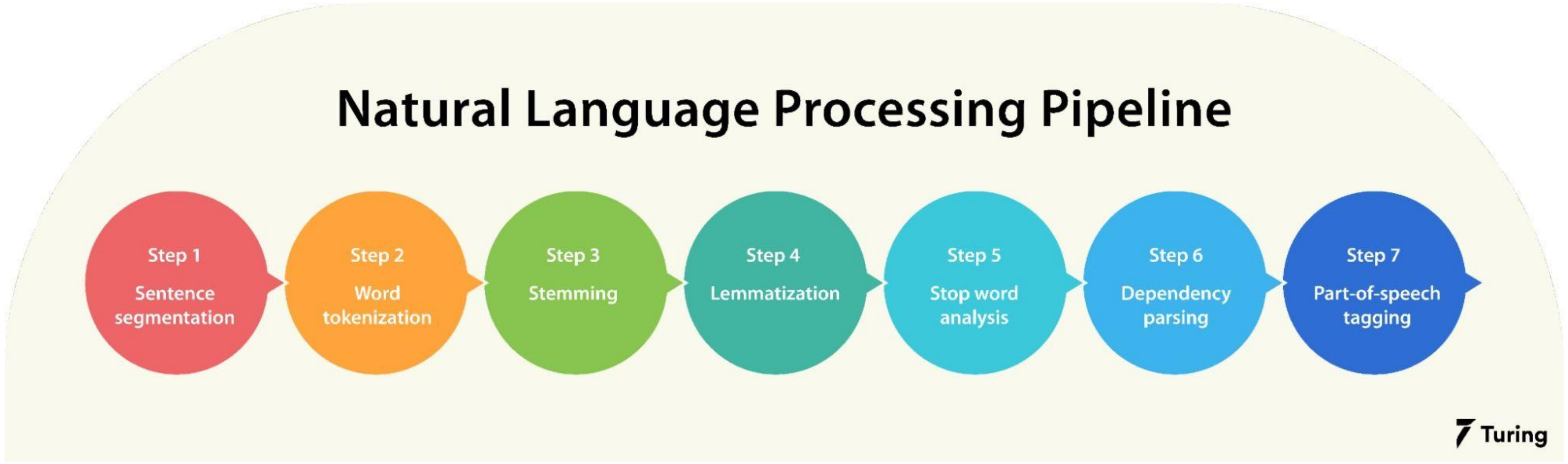
- **Accuracy.** Lemmatization is much more accurate than stemming, as it's able to more precisely determine the lemma of a word.
- **Understanding text.** Lemmatization is useful for tools in NLP like AI chatbots for understanding full sentence input from end users. This is also useful for returning specific search queries.
- **Contextual understanding.** Word-per-word, lemmatization can understand a term based on the contextual use of that word.

Disadvantages

- Computational overhead. Due to the morphological analysis lemmatization conducts on each inflected word.

POS tagging

The pipeline


Natural Language Processing Pipeline
Step 1 Sentence segmentation
Step 2 Word tokenization
Step 3 Stemming
Step 4 Lemmatization
Step 5 Stop word analysis
Step 6 Dependency parsing
Step 7 Part-of-speech tagging
Turing

POS tagging

Part-of-speech (POS) tagging is a popular Natural Language Processing process which refers to categorizing words in a text (corpus) in correspondence with a particular part of speech, depending on the definition of the word and its context.

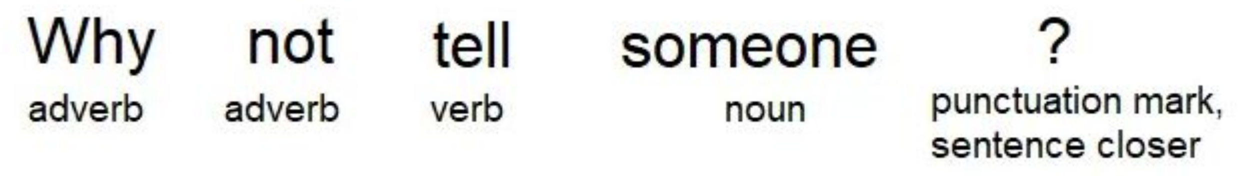
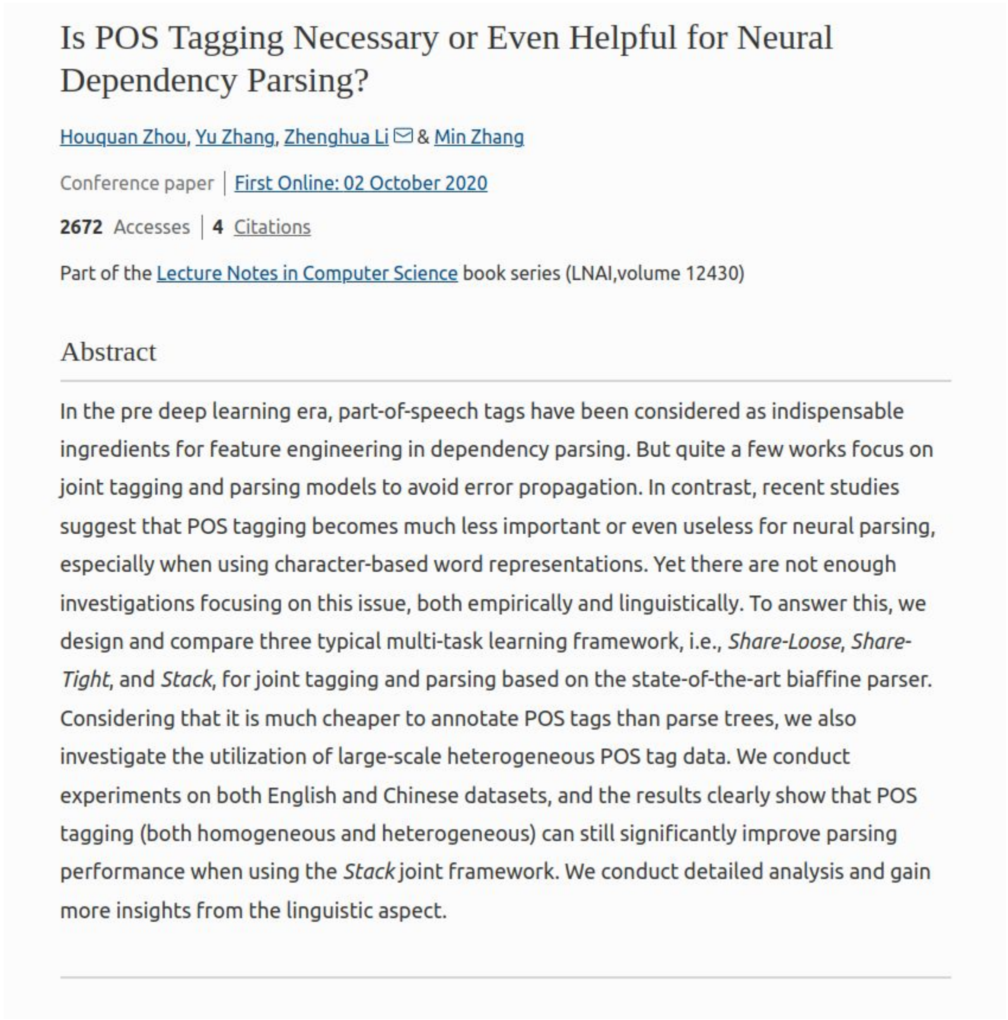


Figure 1: Example of POS tagging (Image by Author)

Syntactic parsing vs POS tagging

We use Part-of-Speech tagging to label tokens in a sentence with their grammatical word categories as part-of-speech tags. Yet, they do not have any grammatical relations between them.

In order to generate the grammatical relations between the tokens, we use linguistic parsers and syntactic dependency parsing is one of them. Via dependency parsing, we create a tree or a graph data structure of a sentence conveying its tokens' grammatical relations. Parsers generally tokenize, tags with POS the sentence for you and then parse. Therefore, we can say parsing a sentence is a further step.



Is POS Tagging Necessary or Even Helpful for Neural Dependency Parsing?

Houquan Zhou, Yu Zhang, Zhenghua Li & Min Zhang

Conference paper | First Online: 02 October 2020

2672 Accesses | **4** Citations

Part of the Lecture Notes in Computer Science book series (LNAI,volume 12430)

Abstract

In the pre deep learning era, part-of-speech tags have been considered as indispensable ingredients for feature engineering in dependency parsing. But quite a few works focus on joint tagging and parsing models to avoid error propagation. In contrast, recent studies suggest that POS tagging becomes much less important or even useless for neural parsing, especially when using character-based word representations. Yet there are not enough investigations focusing on this issue, both empirically and linguistically. To answer this, we design and compare three typical multi-task learning framework, i.e., *Share-Loose*, *Share-Tight*, and *Stack*, for joint tagging and parsing based on the state-of-the-art biaffine parser. Considering that it is much cheaper to annotate POS tags than parse trees, we also investigate the utilization of large-scale heterogeneous POS tag data. We conduct experiments on both English and Chinese datasets, and the results clearly show that POS tagging (both homogeneous and heterogeneous) can still significantly improve parsing performance when using the *Stack* joint framework. We conduct detailed analysis and gain more insights from the linguistic aspect.

Projects

Multilingual dependency parsing

HeadsUp on Parsing

A parser in NLP uses the grammar rules (formal grammar rules) to verify if the input text is valid or not syntactically. The parser helps us to get the meaning of the provided text (like the dictionary meaning of the provided text). As the parser helps us to analyze the syntax error in the text; so, the parsing process is also known as the syntax analysis or the Syntactic analysis.

Simply speaking, parsing in NLP is the process of determining the syntactic structure of a text by analyzing its constituent words based on an underlying grammar (of the language).

Why? Multilingual dependency parsing

Cross-linguistically consistent morphosyntactic annotation Facilitate multilingual research in NLP and linguistics

•Meaningful linguistic analysis across languages

•Syntactic parsing in multilingual settings

•NLP systems for multiple languages

•Facilitate resource-building for new languages

¬Complement – not replace – language-specific schemes

Pressing questions

- How do parsers benefit from pre-trained word embeddings, character models and part-of-speech tags?

•Are the techniques complementary or redundant?

•How do results vary across word frequencies, word categories and languages?

Transition-Based		**Graph-Based**
short sentences		long sentences
short dependencies	**better on**	long dependencies
nouns		verbs
core arguments		main predicates
rich features	**due to**	limited features
greedy decoding		exact decoding

coNLL

http://universaldependencies.org/conll17/

UDapter

https://github.com/ahmetustun/udapter

Discussion