LING 520: Computational Analysis of English

Semester: FALL '16

Instructor: Sowmya Vajjala

Iowa State University, USA

20 September 2016

AACL Conference and Workshops - Comments

- How did the R workshop go? What did you learn?

AACL Conference and Workshops - Comments

- How did the R workshop go? What did you learn?
- How many people in the class are RegEx gurus now?

AACL Conference and Workshops - Comments

- How did the R workshop go? What did you learn?
- How many people in the class are RegEx gurus now?
- Any interesting language processing related talks at AACL?

AACL Conference and Workshops - Comments

- How did the R workshop go? What did you learn?
- How many people in the class are RegEx gurus now?
- Any interesting language processing related talks at AACL?
- Reg Assignment 2: Deadline is on 27th, but do not wait until 26th midnight to ask me for support.

Morphological Parsing

- What is it?: taking in a word and showing its morphological structure as output.

Morphological Parsing

- What is it?: taking in a word and showing its morphological structure as output.
- Examples:

input	output
cars	car + Noun + Plural
caught	catch + Verb + Past
caught	catch + Verb + PastPart
he	he + pronoun + 3rdPerson + Singular

More complex language example

Morphological parsing output for Estonian.

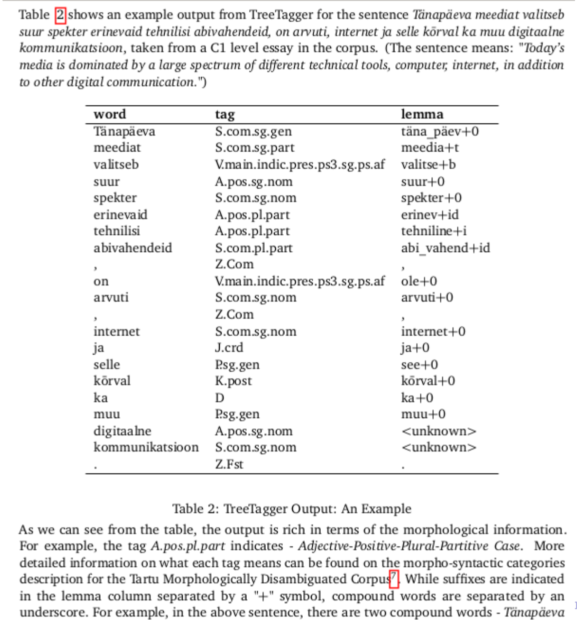

Table 2 shows an example output from TreeTagger for the sentence *Tänapäeva meediat valitseb suur spekter erinevaid tehnilisi abivahendeid, on arvuti, internet ja selle kõrval ka muu digitaalne kommunikatsioon*, taken from a C1 level essay in the corpus. (The sentence means: "*Today's media is dominated by a large spectrum of different technical tools, computer, internet, in addition to other digital communication.*")

word	tag	lemma
Tänapäeva	S.com.sg.gen	täna_päev+0
meediat	S.com.sg.part	meedia+t
valitseb	V.main.indic.pres.ps3.sg.ps.af	valitse+b
suur	A.pos.sg.nom	suur+0
spekter	S.com.sg.nom	spekter+0
erinevaid	A.pos.pl.part	erinev+id
tehnilisi	A.pos.pl.part	tehniline+i
abivahendeid	S.com.pl.part	abi_vahend+id
,	Z.Com	,
on	V.main.indic.pres.ps3.sg.ps.af	ole+0
arvuti	S.com.sg.nom	arvuti+0
,	Z.Com	,
internet	S.com.sg.nom	internet+0
ja	J.crd	ja+0
selle	P.sg.gen	see+0
kõrval	K.post	kõrval+0
ka	D	ka+0
muu	P.sg.gen	muu+0
digitaalne	A.pos.sg.nom	<unknown>
kommunikatsioon	S.com.sg.nom	<unknown>
.	Z.Fst	.

Table 2: TreeTagger Output: An Example

As we can see from the table, the output is rich in terms of the morphological information. For example, the tag *A.pos.pl.part* indicates - *Adjective-Positive-Plural-Partitive Case*. More detailed information on what each tag means can be found on the morpho-syntactic categories description for the Tartu Morphologically Disambiguated Corpus[7]. While suffixes are indicated in the lemma column separated by a "+" symbol, compound words are separated by an underscore. For example, in the above sentence, there are two compound words - *Tänapäeva*

Why get such details?

- Useful to capture spelling variations of words for doing search, especially for morphologically rich languages with lot of inflections.
- Useful to develop good POS taggers and parsers for such languages.
- Useful for creating large dictionaries for spell checking (using edit distance, for example)
- Useful in machine translation, to choose the right translation of a word.
- Useful for speech processing applications as well.

How do we do morphological parsing?

- Language is productive. Some suffixes can be applied to each and every noun or verb or some other class.
- Hence, it is not possible to list all possible forms for all possible words in a language (even in a language without a lot of inflections).
- Solution 1: Copiously create a large set of rules and then write a program implementing those rules.
- Solution 2: Ask linguists to create such parses for a lot of data, and write a program that "learns" the rules based on all those examples.
- Two forms of morphological parsing we discuss today: stemming, lemmatization

Stemming

- In some applications, we don't need a full morphological parse, but we only need a way to group orthographically similar words.
- This can be done by just stripping off word endings using a set of rules. This is called stemming.
- Porter Stemmer is a popular stemming algorithm for English. Originally proposed by Martin Porter in 1980
- There are other stemmers available, and there are stemmers for other languages.

Porter Stemmer

- Rules: from Original 1980 paper http://tartarus.org/~martin/PorterStemmer/def.txt
- NLTK implementation: http://www.nltk.org/_modules/nltk/stem/porter.html
- Note: PorterStemmer just dumbly follows those rules. It does not maintain a lexicon or anything. So, even if you enter a proper name, it stems it.
- In some NLP applications, such limitations do not create any new issues.

Lemmatization

- In some other NLP tasks, grouping words that are from the same root, although they have different surface forms is also important (e.g., goose and geese won't be recognised as one by a stemmer).
- To get the root of a given word, we need to perform lemmatization.
- Lemma of "was" is "be", but stem will be was. Lemma of "believes" is "belief". But stem will be "believ".

Lemmatization: How does it work?

1. What resources do we need?
   - Lists of irregular words and their morphological forms for each POS tag category.
   - Suffix stripping rules like stemming, along with POS tag of the word.
2. For a given word and POS combo, first we look at the list of irregular words for that POS. If the word does not exist in this list, we move to the rules part.

Lemmatization in NLTK

- in NLTK, it uses "Wordnet" to perform lemmatization.
- Wordnet (http://wordnet.princeton.edu/) is much more than a lemmatizer, but has a utility called "Morphy" which does the morphological processing NLTK uses for lemmatization.
- Some information on Morphy: https://wordnet.princeton.edu/man/morphy.7WN.html
- NLTK code: Morphy function definition in http://www.nltk.org/_modules/nltk/corpus/reader/wordnet.html

Practice Exercise

Form into groups of three, and do the following:

1. There is a .txt file on Blackboard, showing how to load various NLTK stemmers and use them.
2. Prepare a test suite of 10 words or so, and study the differences between the outputs of these stemmers.
3. Study the use of WordNetLemmatizer, and how the output varies for the same word based on the POS tag you choose. Test with a few words.
4. If you finish these, find out what other languages does NLTK support for stemming and lemmatizing.

Spend about 20 minutes in doing this exercise, and in the remaining time, give a quick summary of what you discovered.

Bonus Exercises for Enthusiasts

- Learn to use RegExpStemmer in NLTK.
- Start working on Problem Set 3.