

Master Language

Annotation of learner corpus data

Dr. Nathan Vandeweerd

Feb 14th, 2025

Materials and slides available on GitHub

The screenshot shows a GitHub repository page for 'nvandeweerd/ml_seminar'. The repository is public and contains several files and folders:

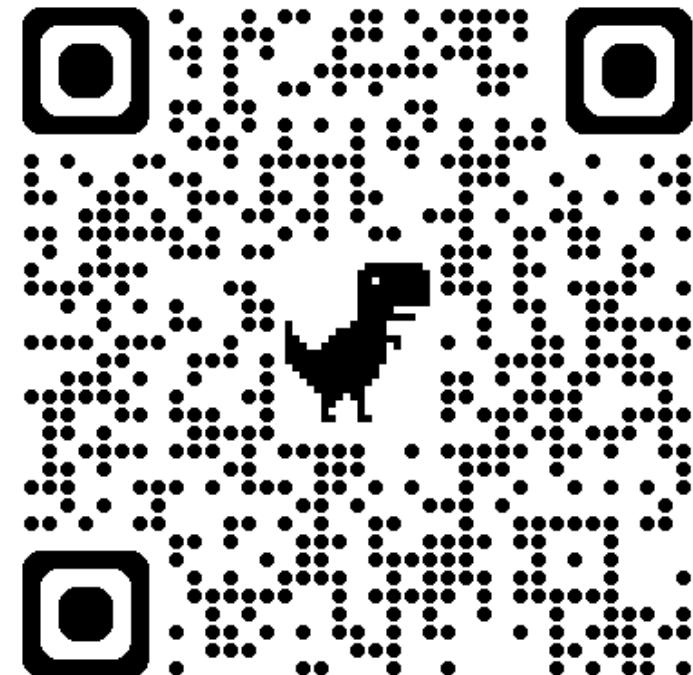
- activities
- data
- slides
- README.html
- README.md
- README

The README file is currently selected. Below the file list, there is a section titled 'Master Language Seminar' which includes an 'Annotation of learner corpus data' section. It lists the dates of the seminar:

- March 15th, 2024 Vrije Universiteit Amsterdam
- February 14th, 2025 Online

At the bottom of the page, it states that the seminar is licensed under CC BY-NC-SA 4.0.

1. Click on **< > Code**.
2. Download ZIP to download all files.



Introduction

About me

III Radboud University Nijmegen

Assistant professor in Language and Communication

Q Research interests:

- Phraseological complexity in L2 French
- Language development during study abroad
- Crowdsourcing language assessment (CLAP)
- Linguistic characteristics of AI vs. L2-produced text



About me

III Radboud University Nijmegen

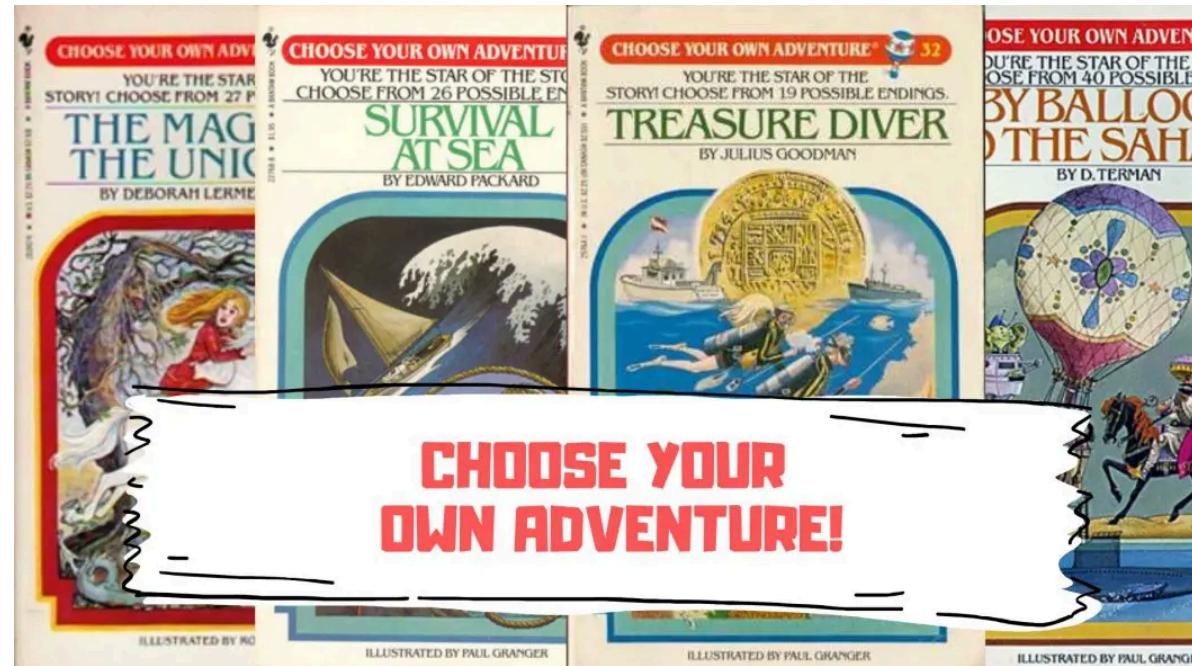
Assistant professor in Language and Communication

Q What makes me happy:

- Eurovision Songcontest
- Baking cookies
- Gym
- Reading



How this workshop will work



Option 1. Webtools/Excel

Option 2. R

Learner corpus annotation in the wild

"Durrant and Schmitt extracted word pairs manually from a raw corpus, while we extracted them automatically from a part-of-speech (POS) tagged corpus. We used CLAWS 73, which has a high degree of accuracy overall (96–97%)" (Granger and Bestgen, 2014, p. 235)

"the progressive forms were extracted from the data with WordSmith Tools 5.0 (Scott, 2008) with the search item *ing and different forms of the verb BE (am, *'m, are, aren'* , *'re, is, *'s, isn'* , was, wasn'* , were, weren'*) as context words within a maximum of five words left." (Meriläinen, 2018, p. 175)

"the present study is based on 1,640,351 words that emerged from the transcribed recordings of the Graded Examination in Spoken English (GESE) which were 166 Sandra Götz conducted at the Trinity College London.⁶ It has been morphologically annotated with the CLAWS tagger (C6 tagset)." (Götz, 2019, p. 165)

Learner corpus annotation in the wild

"Durrant and Schmitt extracted word pairs manually from a raw corpus, while we extracted them automatically from a **part-of-speech (POS) tagged corpus**. We used CLAWS 73, which has a high degree of accuracy overall (96–97%)" (Granger and Bestgen, 2014, p. 235)

"the progressive forms were extracted from the data with WordSmith Tools 5.0 (Scott, 2008) with the **search item *ing** and different forms of the verb BE (am, *'m, are, aren'*, *'re, is, *'s, isn'*, was, wasn'*, were, weren') as context words within a maximum of five words left." (Meriläinen, 2018, p. 175)

"the present study is based on 1,640,351 words that emerged from the transcribed recordings of the Graded Examination in Spoken English (GESE) which were 166 Sandra Götz conducted at the Trinity College London.⁶ It has been **morphologically annotated with the CLAWS tagger (C6 tagset)**." (Götz, 2019, p. 165)

Types of automatic annotation

1. Part of Speech (POS) tagging

Types of automatic annotation

1. Part of Speech (POS) tagging

(1) We_PPIS2 find_VV0 that_CST in_II fact_NN1 these_DD2 people_NN
are_VBR the_AT most_RGT exposed_JJ to_II media_NN not_XX to_TO
mension_VVI the_AT fact_NN1 that_CST there_EX is_VBZ forever_RT
AIDS_NP1 awareness_NN1 campaigns_NN2 launaged_VVN through_RP
out_RP the_AT county_NN1._

(ICLE-TS-NOUN-0005.1)

(van Rooy, 2015: 80)

Research topics

- articles
- morpho-syntactic and syntactic labels
- morphology
- semantic, syntactic and discourse features
- verb valency patterns
- grammatical complexity
- grammatical case
- cohesion and cohesive devices
- stance features
- verb aspect
- adverbs of degree and negation
- verb phrase ellipsis
- formulaic language/phraseology

Types of automatic annotation

1. Part of Speech (POS) tagging

(1) We_PPIS2 find_VV0 that_CST in_II fact_NN1 these_DD2 people_NN
are_VBR the_AT most_RGT exposed_JJ to_II media_NN not_XX to_TO
mension_VVI the_AT fact_NN1 that_CST there_EX is_VBZ forever_RT
AIDS_NP1 awareness_NN1 campaigns_NN2 launaged_VVN through_RP
out_RP the_AT county_NN1._

(ICLE-TS-NOUN-0005.1)

(van Rooy, 2015: 80)

🔍 Research topics

- **articles**
- **morpho-syntactic and syntactic labels**
- **morphology**
- semantic, syntactic and **discourse features**
- **verb** valency patterns
- grammatical complexity
- grammatical **case**
- cohesion and **cohesive devices**
- stance features
- **verb aspect**
- **adverbs** of degree and negation
- **verb** phrase ellipsis
- **formulaic language/phraseology**

Types of automatic annotation

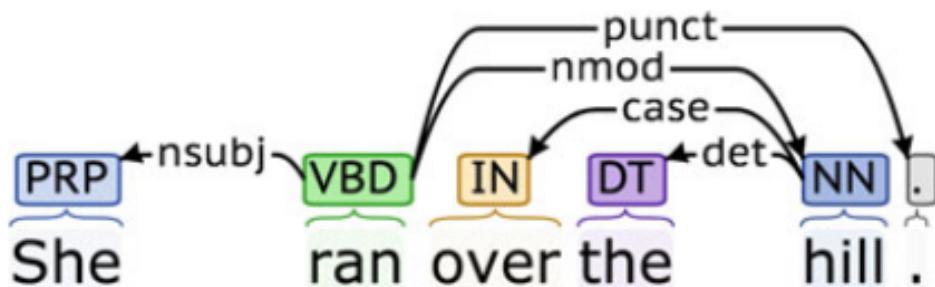
1. Part of Speech (POS) tagging
2. Syntactic parsing

🔍 Research topics

- **articles**
- **morpho-syntactic and syntactic labels**
- **morphology**
- semantic, syntactic and **discourse features**
- **verb** valency patterns
- grammatical complexity
- grammatical **case**
- cohesion and **cohesive devices**
- stance features
- **verb aspect**
- **adverbs** of degree and negation
- **verb** phrase ellipsis
- **formulaic language/phraseology**

Types of automatic annotation

1. Part of Speech (POS) tagging
2. Syntactic parsing



(Newman and Cox, 2021: 32)

Research topics

- articles
- morpho-syntactic and syntactic labels
- morphology
- semantic, syntactic and discourse features
- verb valency patterns
- grammatical complexity
- grammatical case
- cohesion and cohesive devices
- stance features
- verb aspect
- adverbs of degree and negation
- verb phrase ellipsis
- formulaic language/phraseology

Types of automatic annotation

1. Part of Speech (POS) tagging
2. Syntactic parsing
3. Semantic annotation

🔍 Research topics

- articles
- morpho-syntactic and syntactic labels
- morphology
- semantic, syntactic and discourse features
- verb valency patterns
- grammatical complexity
- grammatical case
- cohesion and cohesive devices
- stance features
- verb aspect
- adverbs of degree and negation
- verb phrase ellipsis
- formulaic language/phraseology

Types of automatic annotation

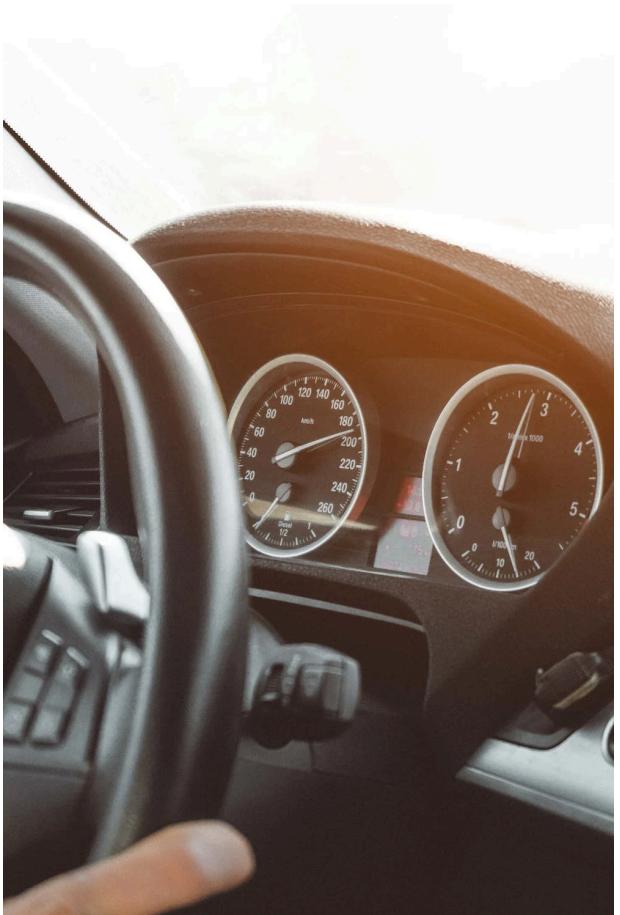
1. Part of Speech (POS) tagging
 2. Syntactic parsing
 3. Semantic annotation
- (6) a. The_Z5 ending_T2- of_Z5 the_Z5 poem_Q3 may_A7+ seem_A8 to_Z5
be_A3+ contradictory_A6.1- because_Z5/A2.2 both_N5 girls_S2.1f
marry_S4 and_Z5 have_A9+ children_S2mf/T3- ;_PUNC thereby_Z5
filling_N5.1+ the_Z5 traditional_S1.1.1 female_S2.1 role_I3.1 ._PUNC
- b. at_T1.1.2[i165.3.1 a_T1.1.2[i165.3.2 time_T1.1.2[i165.3.3

(Newman and Cox, 2021: 35)

Q Research topics

- articles
- morpho-syntactic and syntactic labels
- morphology
- semantic, syntactic and discourse features
- verb valency patterns
- grammatical complexity
- grammatical case
- cohesion and cohesive devices
- stance features
- verb aspect
- adverbs of degree and negation
- verb phrase ellipsis
- formulaic language/phraseology

Why use automatic annotation?



Radboud University 



What are some potential risks of automatic annotation?

Today's session

- Text-preprocessing
- POS-tagging and lemmatization
 - 💻 Hands on activity: *POS-tagging and lemmatization*
- Syntactic annotation
 - 💻 Hands on activity: *Syntactic parsing*
- The reliability of automatic tools on learner data

Text-preprocessing

What do you notice about this text?

I agree that successful people try <e>news</e> things and take risk rather than only doing what they already know how to do well, for these reasons; By trying new things allow you to be curious to know how someone did it and you will find out how to do it too, that way it make you make a research. By trying new things allow you to be positive in your mind and to have a great desire to succeed no matter how difficult is the situation. By trying new things you know that you should be openminded, go through discussion with people who have done the same thing to learn their ways of doing thing, you should meet or have conversation with a lot of these people in order to learn from their experiences. By trying new things you take a big risks, like in "french we say if you don't risk you don't have anything", risk in a good way to take something. We never know if we might succeed or not the only way to do it is to risk. Since we do not lose anything when we risked. As for me doing the same thing every day become boring, i can say is a waste of energy and time, To conclude, people who succeed try new things and take risks rather than only doing what they already know how to do well.



⚠ Ignore spelling mistakes for the time being...

What do you notice about this text?

I agree that successful people try <e>news</e> things and take risk rather than only doing what they already know how to do well, for these reasons; By trying new things allow you to be curious to know how someone did it and you will find out how to do it too, that way it make you make a research. By trying new things allow you to be positive in your mind and to have a great desire to succed no matter how difficult is the situation. By trying new things you no that you should be openminded, go through dissussion with people who have done the same thing to learn their ways of doing thing, you should meet or have conversation with a lot of these people in other to learn from their experiences. By trying new things you take a big risks, like in " french we say if you don't risk you don't have anything", risk in a goog way to takele something. We never know if we might succed or not the only way to do it is to risk. Since we do not loose anything when we risk.ed As for me doing the same thing every day become boring,i can say is a waste of energy and time, To conclude, people who succed try new things and take risks rather than only doing what they already know how to do well.

(Non-exhaustive) list of things that can cause issues for automatic tools:

- (Inconsistent) file encoding
- Spacing
 - Lack of space between words
 - Unnecessary space between words
 - Double space between words
- 'Stylized' apostrophes or quotation marks
- Accented characters (e.g., à)
- Special characters (e.g., *, %, |)
- Inconsistent spelling rules (e.g., email/e-mail)
- Coding schemes (e.g., XML)

What do you notice about this text?

I agree that successful people try <e>news</e> things and take risk rather than only doing what they already know how to do well, for these reasons; By trying new things allow you to be curious to know how someone did it and you will find out how to do it too, that way it make you make a research. By trying new things allow you to be positive in your mind and to have a great desire to succed no matter how difficult is the situation. By trying new things you no that you should be openminded, go through dissussion with people who have done the same thing to learn their ways of doing thing, you should meet or have conversation with a lot of these people in other to learn from their experiences. By trying new things you take a big risks, like in " french we say if you don't risk you don't have anything", risk in a goog way to takele something. We never know if we might succed or not the only way to do it is to risk. Since we do not loose anything when we risk.ed As for me doing the same thing every day become boring,i can say is a waste of energy and time, To conclude, people who succed try new things and take risks rather than only doing what they already know how to do well.

(Non-exhaustive) list of things that can cause issues for automatic tools:

- (Inconsistent) file encoding
- Spacing
 - Lack of space between words
 - Unnecessary space between words
 - Double space between words
- 'Stylized' apostrophes or quotation marks
- Accented characters (e.g., à)
- Special characters (e.g., *, %, |)
- Inconsistent spelling rules (e.g., email/e-mail)
- Coding schemes (e.g., XML)



! Importance of **knowing your corpus...**

Methods of text cleaning/preprocessing

- +
- Time intensive
-
- Replicable
- 1. Manually
- 2. Semi-manually using regular expressions (e.g., in a text editor)
- 3. Semi-automatically using a script with regular expressions (e.g., in R or python)

- Time intensive
- +
- Replicable

Regular expressions (RegEx)

Special patterns that allow you to search for specific sequences.

Examples:

\w: Returns a word character (A-Z, a-z, _)

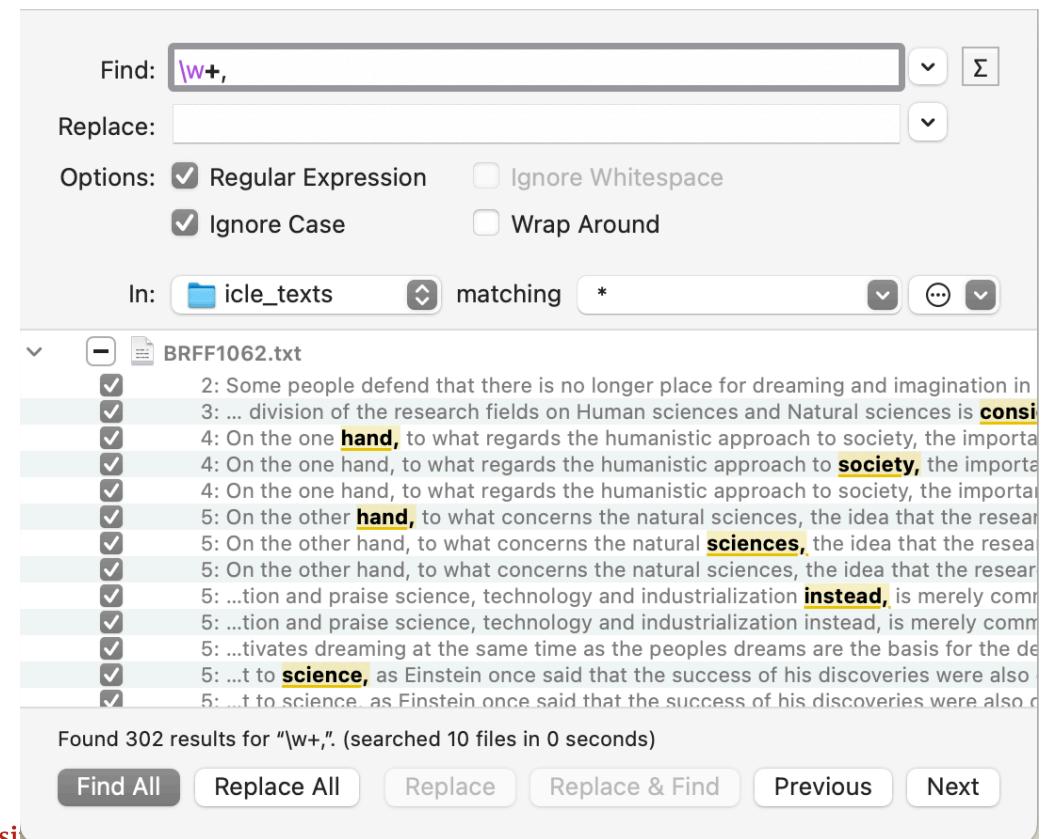
\s: Returns a space character

+: Returns one or more of the previous character

? : Returns zero or more of the previous character

.: Returns any single character

See [here](#) for a website where you can test regular expressions.



Regular expressions (RegEx)

Special patterns that allow you to search for specific sequences.

Examples:

\w: Returns a word character (A-Z, a-z, _)

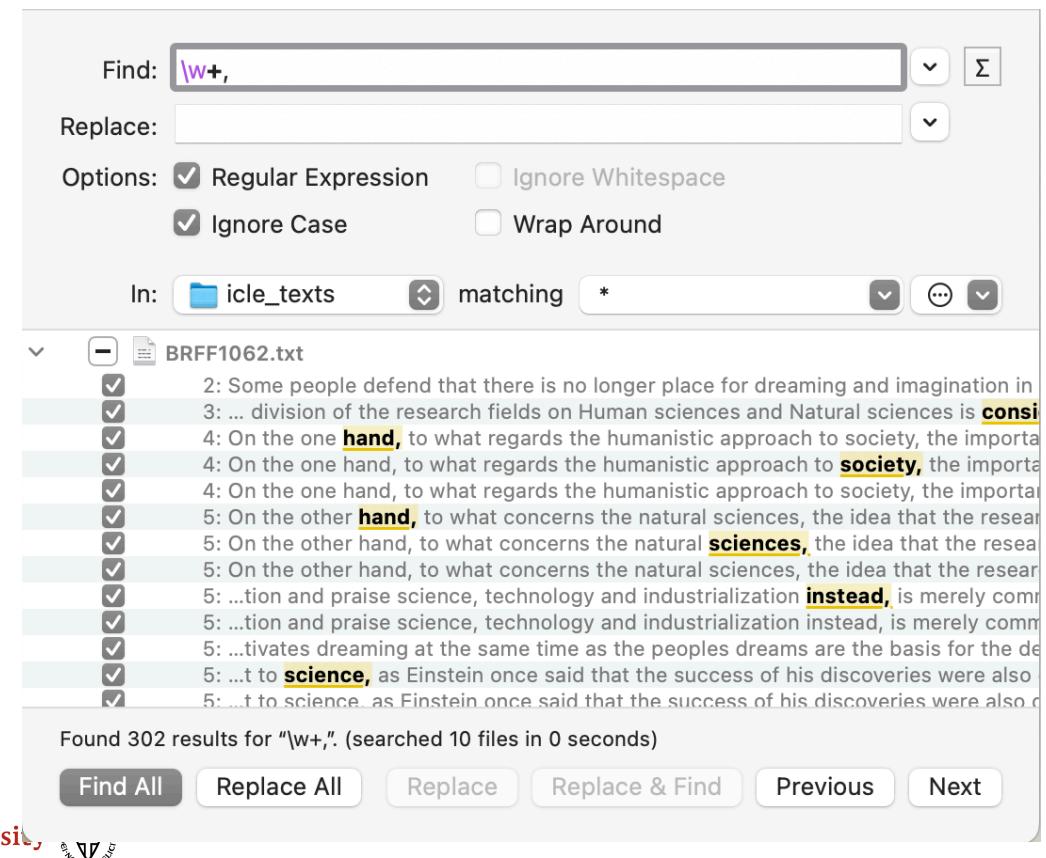
\s: Returns a space character

+: Returns one or more of the previous character

? : Returns zero or more of the previous character

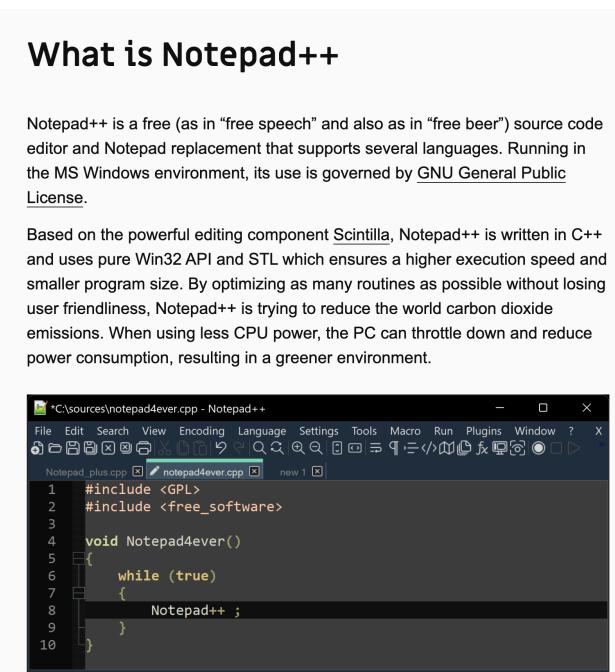
.: Returns any single character

Try this at home: What regular expression could be used to remove the 'headers' from the ICLE texts (data/icle_texts)? (e.g., <ICLE-BR-FF-0062.1>). Be careful not to remove anything else! (Answers at the end of the slides).



Notepad++ (PC) / Textmate (Mac)

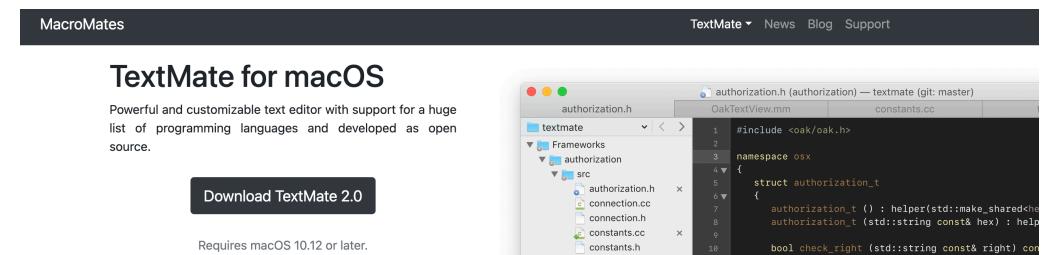
Notepad ++



The screenshot shows the Notepad++ application window. On the left, there's a sidebar with a logo of a cartoon character holding a pencil, the text "Current Version 8.5.4", and a navigation menu with links to Home, Download, News, Online Help, Resources, RSS, Donate, and Author. A green advertisement for "think-cell" is visible at the bottom left. The main area shows a code editor with two tabs open: "Notebook_plus.cpp" and "notepad4ever.cpp". The code in "notepad4ever.cpp" is as follows:

```
1 #include <GL>
2 #include <free_software>
3
4 void Notepad4ever()
5 {
6     while (true)
7     {
8         Notepad++ ;
9     }
10 }
```

Textmate



The screenshot shows the Textmate application window. At the top, there's a menu bar with "TextMate ▾ News Blog Support". The main area has a title "MacroMates" and a sub-section "TextMate for macOS". Below that, it says "Powerful and customizable text editor with support for a huge list of programming languages and developed as open source." A "Download TextMate 2.0" button is present. To the right, there's a file browser showing a directory structure for "textmate" with files like "authorization.h", "connection.h", "constants.cc", and "constants.h". A preview pane shows some C++ code.

Multiple Carets

Making multiple changes at once, swapping pieces of code, and a lot more is made trivial with TextMate's easy way to add multiple insertion points.

Scoped Settings

One file mixing languages? Projects using different build systems? Third party code with different formatting preferences? TextMate can handle it all by associating detailed scope selectors with key shortcuts, settings, etc.

File Search

Select what you want to search, what you want to search for, and TextMate will present the results in a way that makes it easy to jump between matches, extract matched text, or preview desired replacements.

Commands

The UNIX underpinnings of macOS allows custom actions to be written in any language that can work with stdin, stdout, and environment variables, and for complex interactions TextMate exposes both WebKit and a dialog framework for Mac-native or HTML-based interfaces.

Version Control

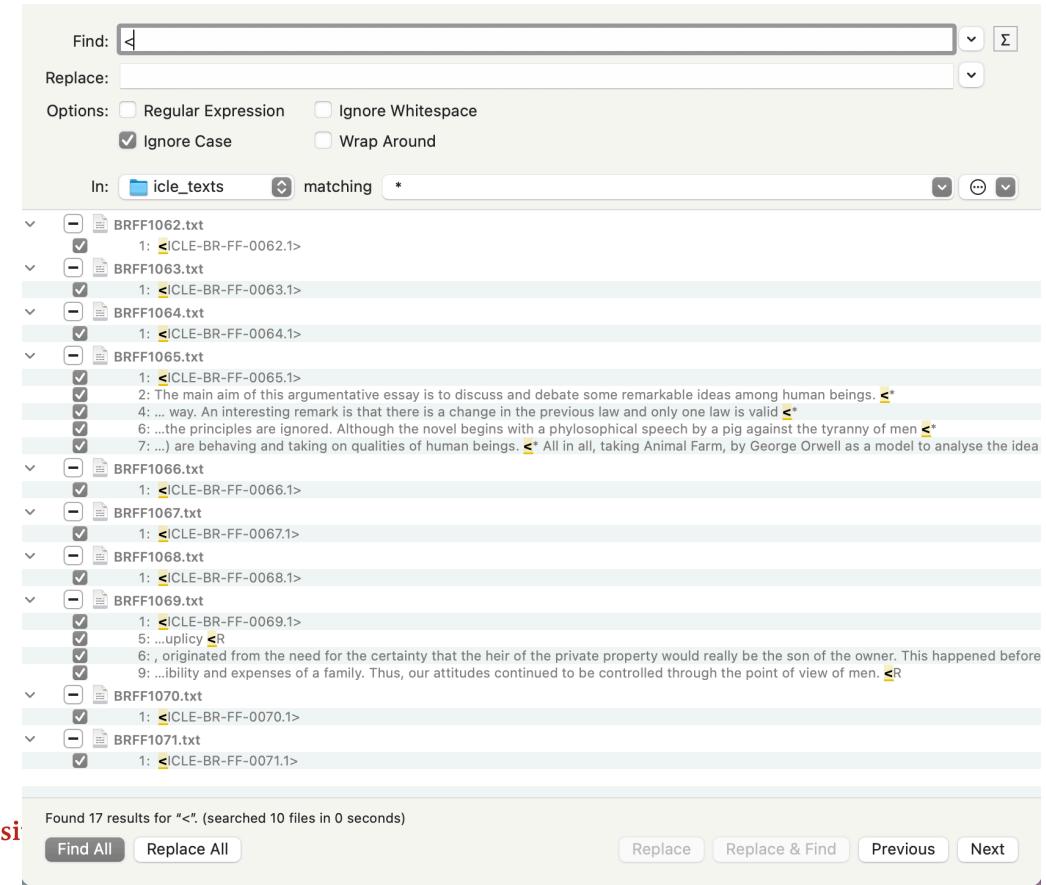
See what files have changes in the file browser view, what lines have changes in the editor view, bring up a diff of the current file's changes, commit a subset, TextMate supports it all for all the major version control systems.

Snippets

Commonly used pieces of text or code can be turned into snippets with placeholders, transformations, and more, to have them adapt according to the context in which they are used.

Textmate (Mac)

A screenshot of the Textmate application interface. On the left, a code editor window titled "BRFF1062.txt — icle_texts (git: main)" displays a large block of text. The text discusses the relationship between dreaming, imagination, science, technology, and industrialization. It mentions the importance of researchers from various fields and the need to examine the principles behind the end of dreaming and imagination in the modern world. The text is heavily annotated with red dotted underlines. On the right, a file browser window titled "icle_texts" shows a list of files: BRFF1062.txt, BRFF1063.txt, BRFF1064.txt, BRFF1065.txt, BRFF1066.txt, BRFF1067.txt, BRFF1068.txt, BRFF1069.txt, BRFF1070.txt, and BRFF1071.txt.



R

```
library(stringr); library(dplyr)

text <- "This is a sentence that 's got a stylized apostrophe."
text %>%
  # replace stylized apostrophes
  str_replace_all("'", "") %>%
  # remove spaces before apostrophes
  # if they are:
  # - preceded by the beginning of a string, space or final punctuation
  # - followed by only one or two letters (e.g., don't, they're)
  # - and then followed by either a space, final punctuation or end of string
  str_replace_all("[\\s\\.\\?\\!^]([^\s]{1,2}[\\s\\.\\?\\!$])", "'\\1")  

## [1] "This is a sentence that's got a stylized apostrophe."
```



Note that you need double slashes for special or escape characters in R (\\").

With both approaches

⚠ Test (and re-test) your pre-processing pipeline!

- Be careful of inadvertent changes (especially when using regular expressions)
E.g. *don't* vs. *he said 'don't worry'*
- Be aware the different tools require different approaches (e.g., contractions separate or apart)
- Never edit the original corpus files!
- Keep track of any changes you make

```
text <- "'He 'll be comin' round the 'mountain' when he comes,' I said."  
  
text %>%  
  str_replace_all("'", "") %>%  
  str_replace_all("[\\s\\.\\?\\!^]([^\s]{1,2}[\\s\\.\\?\\!$])", '\\1')  
  
## [1] "'He'll be comin' round the 'mountain' when he comes,' I said."
```

So why is pre-processing so important?

```
[1] "I"                                "agree"
[3] "that"                             "successful"
[5] "people"                           "try"
[7] "&lt;e&gt;news&lt;/e&gt;" "things"
[9] "and"                              "take"
[11] "risk"                            "rather"
[13] "than"                            "only"
[15] "doing"                           "what"
[17] "they"                            "already"
[19] "know"                            "how"
[21] "to"                               "do"
[23] "well, for"                      "these"
[25] "reasons;"                         ""
[27] "By"                               "trying"
[29] "new"                             "things"
```



Garbage in...garbage out

Tokenization

Token: "the smallest unit of a corpus" (Krause, Lüdeling, Odebrecht, and Zeldes, 2012: 2)

= words, numbers, punctuation marks, quotation marks etc.

= syllable, phoneme, etc...

Easiest method (for English):

- split tokens at spaces
- split sentences at periods (.), exclamation marks (!) or question marks (?)

What problems do you see with this method?

Potential problems:

- clitics (isn't, ain't)
- missing whitespace
- periods (etc., U.S.A., fig.)
- ordinal numbers
- multiword expressions (New York-based, 10 000, as well as)
- word-internal punctuation (relationship(s), "Rambo"-type)
- (de)hyphenation (preprocessing vs. pre-processing)
- quoted speech ("You still don't have an accountant?" Ellis said.)
- ideographic languages (e.g., Chinese, Japanese)

❑ For more information about tokenization see Zeldes (2020) and Schmid (2008).

Tokenization

The upside: Most automatic tools include a tokenization step.

- Usually 'rules'-based
- Important to check yourself how this will influence your data/analyses

POS-tagging and lemmatization

Overview

Each word in the corpus is 'tagged' (labelled) with information about its grammatical category.

🔧 Under the hood:

1. All tokens with unambiguous POS labels are assigned tags (e.g., on the basis of a dictionary)
2. Contextual features (e.g., surrounding tags, morphological endings) used in a statistical model to predict the tags of ambiguous items

She ♦ pronoun
sells ♦ verb
seashells ♦ noun
by ♦ preposition
the ♦ determiner
seashore ♦ noun
. ♦ punctuation

(Kyle, 2021: 6)

Tagsets for English

- CLAWS (Constituent Likelihood Automatic Word-tagging System)
 - CLAWS 5 = 60 tags
 - CLAWS 7 = 160 tags
- PENN Treebank Tagset
- BNC Tagset
- Universal POS tags

Different tagsets, (subtly) different theories of grammar

Table 2.1 Four tagging solutions for English *rid*

	<i>I am now completely rid of such things</i>	<i>You are well rid of him</i>	<i>I got rid of the rubbish</i>
CLAWS7 tagger ^a	Past participle	Past participle	Past participle
Infogistics ^b	Verb base	Verb base	Past participle
FreeLing ^c	Adjective	Verb base	Past participle
(Brill-based) GoTagger ^d	Adjective	Adjective	Adjective

(Newman and Cox, 2021: 21)

💡 Select the tagset that is right for your data/research question.

Example

(1) We_PPIS2 find_VV0 that_CST in_II fact_NN1 these_DD2 people_NN are_VBR the_AT most_RGT exposed_JJ to_II media_NN not_XX to_TO mension_VVI the_AT fact_NN1 that_CST there_EX is_VBZ forever_RT AIDS_NP1 awareness_NN1 campaigns_NN2 launged_VVN through_RP out_RP the_AT county_NN1._

(ICLE-TS-NOUN-0005.1)

(2) We find that in fact these people are the most exposed to media not to mension the fact that there is forever AIDS awareness campaigns launged through out the county.

Verbs: launged_VVD, find_VV0, is_VBZ, are_VBR

Nouns: media_NN, fact_NN1, AIDS_NN1, county_NN1, awareness_NN1, people_NN, mension_NN1, campaigns_NN2, fact_NN1

= CLAWS C7 Tagset

(van Rooy, 2015: 80-81)

② What does the tag PPIS2 refer to?

③ What do you notice about the learner errors? (e.g., 'launged', 'mension', 'is awareness campaigns')

Lemmatization

Lemma: "a 'base form', which provides a level of abstraction from any inflection that might appear in the original orthographic word."

(Newman and Cox, 2021: 29)

- (2) We find that in fact these people are the most exposed to media not to mention the fact that there is forever AIDS awareness campaigns launched through out the county.

we find that in fact these people be the most expose to medium not to mention the fact that there be forever AIDS awareness campaign launched through out the county .

Q Why might this be useful?

Example

	##	idx	sntc	token	tag	lemma	lttr	wclass
##	1	1	1	We	PP	we	2	pronoun
##	2	2	1	find	VBP	find	4	verb
##	3	3	1	that	IN	that	4	preposition
##	4	4	1	in	IN	in	2	preposition
##	5	5	1	fact	NN	fact	4	noun
##	6	6	1	these	DT	these	5	determiner
##	7	7	1	people	NNS	people	6	noun
##	8	8	1	are	VBP	be	3	verb
##	9	9	1	the	DT	the	3	determiner
##	10	10	1	most	RBS	most	4	adverb
##	11	11	1	exposed	VBN	expose	7	verb
##	12	12	1	to	TO	to	2	to
##	13	13	1	media	NNS	medium	5	noun
##	14	14	1	not	RB	not	3	adverb
##	15	15	1	to	TO	to	2	to
##	16	16	1	mension	NN	<unknown>	7	noun
##	17	17	1	the	DT	the	3	determiner
##	18	18	1	fact	NN	fact	4	noun
##	19	19	1	that	IN	that	4	preposition
##	20	20	1	there	EX	there	5	existential

Webtools

TreeTagger

Online TreeTagger

Annotate your texts with part-of-speech and lemma information using [TreeTagger](#).

Type a text Upload a file

Text to process*
Type your text here or copy/paste it

Language of your text*

CLAWS

UCREL API

Free CLAWS web tagger

Our free web tagging service offers access to the latest version of the tagger, CLAWS4, which was used to POS tag c.100 million words of the original [British National Corpus \(BNC1994\)](#), the [BNC2014](#), and all the English corpora in Mark Davies' [BYU corpus server](#). You can choose to have output in either the smaller [C5 tagset](#) or the larger [C7 tagset](#).

[CLAWS POS tagger](#) | [Obtaining a licence](#) | [Tagging service](#)

If you would like to use our free WWW tagger, please complete the form below. You can enter up to 100,000 words of English running text. If you enter more, it will be cut off at the word limit. [Input format guidelines](#) are available. To tag the text you have entered click the button below the form.

Select tagset: C5 C7

Select output style: Horizontal Vertical Pseudo-XML

Type (or paste) your text to be tagged into this box.

R

```
library(spacyr)  
spacy_initialize(model = "en_core_web_sm")  
text <- "She ran over the hill."  
spacy_parse(text)
```

⚠ Note: For this to work, spaCy must be installed locally on your computer but this can be done within the spaCy package using the `spacy_install()` function.

💡 See [this vignette](#) for more information about the `spacyr` package.

Activity: POS-tagging

Webtools/Excel Option

Open activity_01_pos-tagging.docx and follow the instructions.

R Option

Open activity_01_pos-tagging.R and follow the instructions.

💡 Remember that all materials and slides available on [GitHub](#).

1. Click on [`< > Code`](#).
2. [Download ZIP](#) to download all files.

Activity: POS-tagging

Webtools/Excel Option

Open `activity_01_pos-tagging.docx` and follow the instructions.

R Option

Open `activity_01_pos-tagging.R` and follow the instructions.

💡 Remember that all materials and slides available on [GitHub](#).

1. Click on [`< > Code`](#).
2. [Download ZIP](#) to download all files.

For R-users

- Make sure you have installed all required R packages (see `set-up.R`) and that you have a spaCy installation on your computer beforehand
- If not: please use the `.rds` objects to load the spaCy-generated data in (installing spaCy can take a long time)

Syntactic parsing

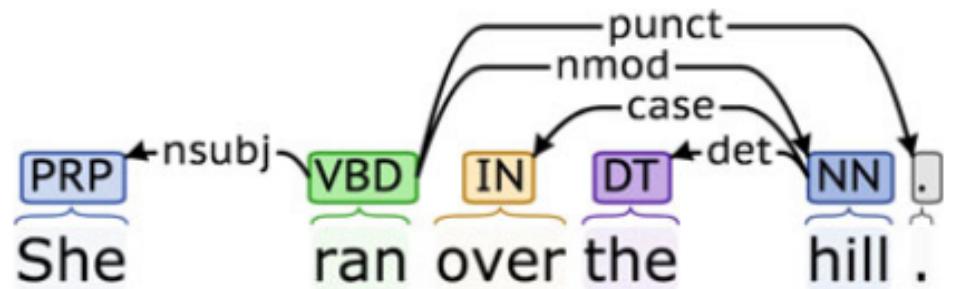
Overview

Labels of the syntactic connections between words (heads and dependents)

🔧 Under the hood:

1. Texts are POS-tagged.
2. POS tags used in conjunction with phrase-structure rules (generated from training algorithms on large corpora) to generate several possible *parse trees* for each sentence.
3. Statistical or machine learning algorithms are used to select the most probable parse tree for a given sentence.

(Kyle, 2021: 7)



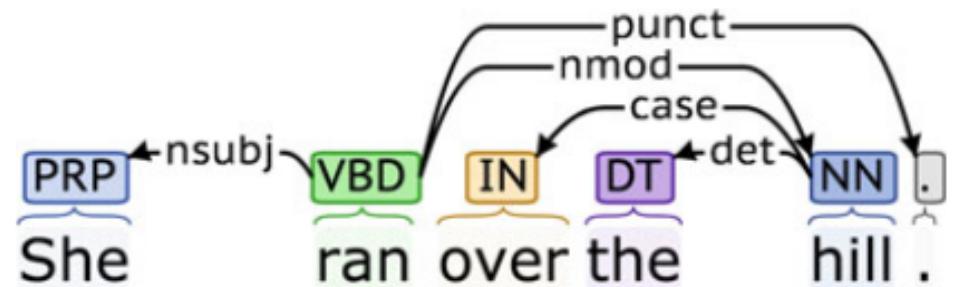
Overview

Labels of the syntactic connections between words (heads and dependents)

🔧 Under the hood:

1. Texts are POS-tagged.
2. POS tags used in conjunction with phrase-structure rules (generated from training algorithms on large corpora) to generate several possible *parse trees* for each sentence.
3. Statistical or machine learning algorithms are used to select the most probable parse tree for a given sentence.

(Kyle, 2021: 7)



(3) Parse of *She ran over the hill.*
(ROOT
(S
 (NP (PRP She))
 (VP (VBD ran))
 (PP (IN over))
 (NP (DT the) (NN hill))))
 (.))))

Dependency models for English

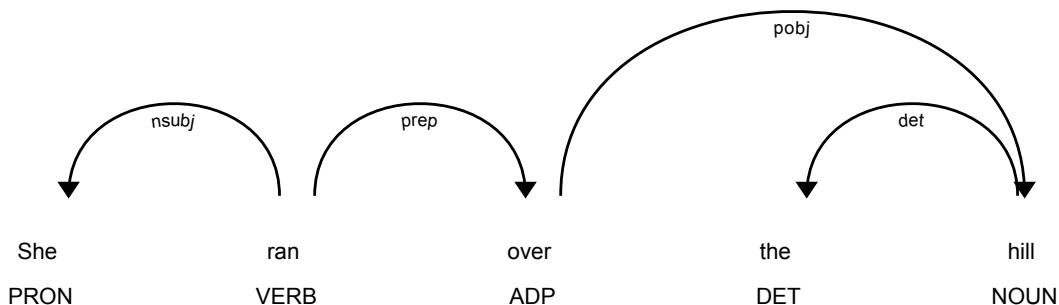
- Stanford CoreNLP
- spaCy
- Universal dependencies

Different models, (subtly) different theories of grammar

CoNLL

= Conference on Natural Language Learning

doc_id	sentence_id	token_id	token	lemma	pos	head_token_id	dep_rel
1	text1	1	1	She	she	PRON	2 nsubj
2	text1	1	2	ran	run	VERB	2 ROOT
3	text1	1	3	over	over	ADP	2 prep
4	text1	1	4	the	the	DET	5 det
5	text1	1	5	hill	hill	NOUN	3 pobj
6	text1	1	6	.	.	PUNCT	2 punct



Webtools

Hugging Face spaCy visualizer

The screenshot shows the Hugging Face spaCy pipeline visualizer interface. On the left, there's a sidebar with sections for 'spaCy', 'Explore trained spaCy pipelines', 'Model' (set to 'en_core_web_sm'), 'Pipeline info' (describing 'en_core_web_sm v3.4.0'), and 'Visualizers' (with options for 'parser', 'ner', 'similarity', and 'tokens'). The main area displays the input sentence: 'Apple is looking at buying U.K. startup for \$1 billion'. Below the input is a 'Dependency Parse & Part-of-speech tags' section. A dependency tree diagram is shown with arrows indicating relationships between words: 'Apple' (PROPN) is the subject of 'is' (AUX), which is part of the verb phrase 'looking at'. 'buying' (VERB) is the object of 'buying'. 'U.K.' (NOUN) and 'startup' (NOUN) are objects of 'buying'. 'for' (ADP) and '\$1 billion' (NUM) are objects of 'buying'. The POS tags are: Apple (PROPN), is (AUX), looking (VERB), at (ADP), buying (VERB), U.K. (NOUN), startup (NOUN), for (ADP), \$1 (NUM), billion (NUM).

LINDAT UDPipe

The screenshot shows the LINDAT UDPipe service interface. At the top, there's a navigation bar with links for Catalog, Repository, Education, Projects, Tools, Services, and About. Below the navigation is a sub-navigation bar for 'LINDAT/CLARIN / Services / UDPipe'. The main content area is titled 'UDPipe' and includes sections for 'About', 'Run', and 'REST API Documentation'. It provides information about UDPipe being a trainable pipeline for tokenization, tagging, lemmatization, and dependency parsing of CoNLL-U files. It mentions that UDPipe is language-agnostic and can be trained given annotated data in CoNLL-U format. Trained models are provided for nearly all UD treebanks. UDPipe is available as a binary for Linux/Windows/OS X, as a library for C++, Python, Perl, Java, C#, and as a web service. Third-party R CRAN package also exists. UDPipe is a free software distributed under the Mozilla Public License 2.0 and the linguistic models are free for non-commercial use and distributed under the CC BY-NC-SA license, although for some models the original data used to create the model may impose additional licensing conditions. UDPipe is versioned using Semantic Versioning. Copyright 2017 by Institute of Formal and Applied Linguistics, Faculty of Mathematics and Physics, Charles University, Czech Republic. Description of the available methods is available in the API Documentation and the models are described in the UDPipe 2 models list and UDPipe 1 models list.

The 'Service' section indicates that the service is freely available for testing. It requires respecting the CC BY-NC-SA licence of the models - explicit written permission of the authors is required for any commercial exploitation of the system. If you use the service, you agree that data obtained by us during such use can be used for further improvements of the systems at UFAL. All comments and reactions are welcome.

The 'Actions' section allows selecting 'Tag and Lemmatize' and 'Parse'. The 'Advanced Options' section includes 'Input Text' and 'Input File' fields. The 'Model' dropdown is set to 'UD 2.1.0 (docs)' with the English model selected ('english-ud-2.1.0-241121').

(At home) Activity: Syntactic parsing

Webtools/Excel Option

Open `activity_02_parsing.docx` and follow the instructions.

R Option

Open `activity_02_parsing.R` and follow the instructions.

💡 Remember that all materials and slides available on [GitHub](#).

1. Click on `< > Code`.
2. [Download ZIP](#) to download all files.

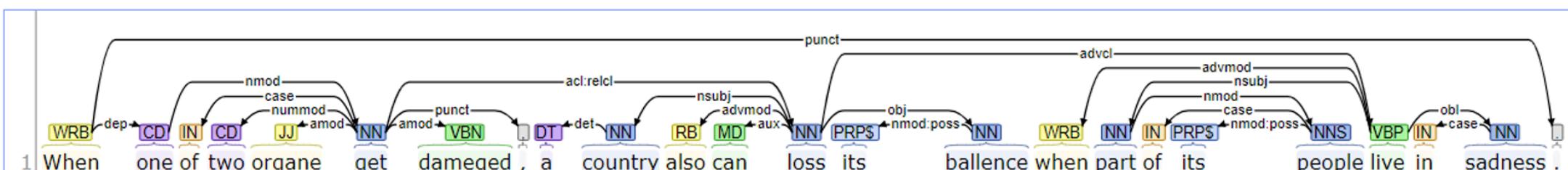
The reliability of automatic tools

The effect of learner errors

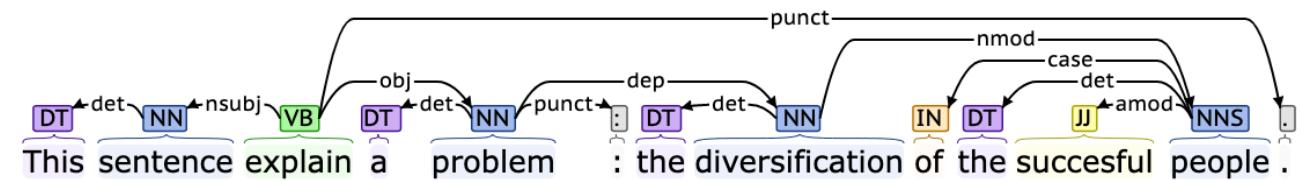
- (4) a. ... like body that will **loss** its **ballence** when one of two **organe** get **dameged**, a country also can **loss** its **ballence** when part of its people live in sadness.
- b. ... **peoples** who I met is not good ...
- c. So when I **admit** to korea university, I decide what i find my own way.
- d. Also, the people in it very friendly.

(Ragheb and Dickinson, 2012)

What problems do you see here?



Not all errors are equally as problematic



(Ragheb and Dickinson, 2012)

Evaluating the *reliability* of automatic tools

Confusion matrix: A tabulation of the agreement between manual and automatic annotation.

Example:

```
##      unit human computer
## 1    apple   noun     noun
## 2      run   verb     noun
## 3     cake   noun     verb
## 4   coffee   noun     noun
## 5    drink   verb     noun
## 6    swim   verb     verb
## 7     tart   noun     verb
## 8    walk   verb     noun
## 9     ice   noun     noun
## 10    pen   noun     noun
```

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Evaluating the *reliability* of automatic tools

Confusion matrix: A tabulation of the agreement between manual and automatic annotation.

		Reference	
		Prediction	noun
		noun	4
		verb	3
			1

Example:

```
##      unit human computer
## 1    apple   noun    noun
## 2     run    verb    noun
## 3    cake   noun    verb
## 4   coffee   noun    noun
## 5   drink    verb    noun
## 6    swim    verb    verb
## 7    tart   noun    verb
## 8   walk    verb    noun
## 9     ice   noun    noun
## 10    pen   noun    noun
```

Evaluating the *reliability* of automatic tools

Precision: the extent to which the retrieved objects in a query are correctly tagged

Recall/Sensitivity: the extent to which the objects matching the query retrieve all the target objects in the corpus

F-score/F1-score the balance between precision and recall (what is normally reported)

Evaluating the *reliability* of automatic tools

Precision: the extent to which the retrieved objects in a query are correctly tagged

$$\frac{TP}{(TP + FP)}$$

Recall/Sensitivity: the extent to which the objects matching the query retrieve all the target objects in the corpus

$$\frac{TP}{(TP + FN)}$$

F-score/F1-score the balance between precision and recall (what is normally reported)

$$\frac{2 * (P * R)}{P + R}$$

Reliability of automatic annotation for fsca tool
(Vandeweerd, 2021: 265)

Unit	Precision	Recall	F-score
Sentences	0.96	0.97	0.97
Clauses	0.75	0.72	0.74
Dependent Clauses	0.63	0.58	0.60
Coordinated Clauses	0.54	0.51	0.53
T-units	0.83	0.82	0.83
Noun Phrases	0.84	0.84	0.84
Verb Phrases	0.78	0.78	0.78

Webtools

Confusion Matrix Online Calculator

Confusion Matrix About Measures Example Contact

Confusion Matrix

Online Calculator

	True Positive	True Negative
Predicted Positive	True Positive	False Positive
Predicted Negative	False Negative	True Negative

Calculate

R

```
library(caret)  
confusionMatrix(data, reference)
```

💡 See [this vignette](#) for more information about the caret package.

```
# Confusion Matrix and Statistics  
#  
#           Reference  
# Prediction   M   R  
#           M 21  7  
#           R  6 17  
#  
#           Accuracy : 0.745  
#                     95% CI : (0.604, 0.857)  
# No Information Rate : 0.529  
# P-Value [Acc NIR] : 0.00131  
#  
#           Kappa : 0.487  
#  
# Mcnemar's Test P-Value : 1.00000  
#  
#           Sensitivity : 0.778  
#           Specificity : 0.708  
# Pos Pred Value : 0.750  
# Neg Pred Value : 0.739  
# Prevalence : 0.529  
# Detection Rate : 0.412  
# Detection Prevalence : 0.549  
# Balanced Accuracy : 0.743  
#  
# 'Positive' Class : M
```

Final remarks

Recap

1. What are some types of automatic annotation that can be applied to learner texts?
2. Why is it important to clean and pre-process your texts before using automatic tools?
3. What is POS-tagging?
 - What are some POS tagging tools?
4. What is syntactic parsing?
 - What are some tools for syntactic parsing?
5. How can you evaluate the reliability of automatic annotation tools?

Q & Eh?

Further reading

Kyle, K. (2021). Natural language processing for learner corpus research. *International Journal of Learner Corpus Research*, 7(1), 1–16.

Murakami, A., Thompson, P., Hunston, S., & Vajn, D. (2017). "What is this corpus about?": Using topic modelling to explore a specialised corpus. *Corpora*, 12(2), 243–277.

Newman, J., & Cox, C. (2021). Corpus annotation. In M. Paquot & S. Th. Gries (Eds.), *A practical handbook of corpus linguistics* (pp. 25–48). Springer.
<https://doi.org/10.1007/978-3-030-46216-1>

Schmid, H. (2008). Tokenizing and part-of-speech tagging. In A. Lüdeling & M Kytö (Eds.) *Corpus linguistics: An international handbook*. de Gruyter.

van Rooy, B. (2015). Annotating learner corpora. In S. Granger, G. Gilquin, & F. Meunier (Eds.), *The Cambridge handbook of learner corpus research* (pp. 79–106). Cambridge University Press. <https://doi.org/10.1017/CBO9781139649414.005>

Zeldes, A. (2021). Corpus Architecture. In M. Paquot & S. Th. Gries (Eds.), *A practical handbook of corpus linguistics* (pp. 49–73). Springer.
<https://doi.org/10.1007/978-3-030-46216-1>

Solutions

Regex Solution

How to find...<ICLE-BR-FF-0062.1>

<\w{4}-\w{2}-\w{2}-\w{4}\.\.\d>

<[^>]+>

▼ / <\w{4}-\w{2}-\w{2}-\w{4}\.\.\d> / gm
< matches the character < with index 60₁₀ (3C₁₆ or 74₈) literally
(case sensitive)
▼ \w matches any word character (equivalent to [a-zA-Z0-9_])
 {4} matches the previous token exactly 4 times
- matches the character - with index 45₁₀ (2D₁₆ or 55₈) literally
(case sensitive)
▼ \w matches any word character (equivalent to [a-zA-Z0-9_])
 {2} matches the previous token exactly 2 times
- matches the character - with index 45₁₀ (2D₁₆ or 55₈) literally
(case sensitive)
▼ \w matches any word character (equivalent to [a-zA-Z0-9_])
 {2} matches the previous token exactly 2 times
- matches the character - with index 45₁₀ (2D₁₆ or 55₈) literally
(case sensitive)
▼ \w matches any word character (equivalent to [a-zA-Z0-9_])
 {4} matches the previous token exactly 4 times
\. matches the character \. with index 46₁₀ (2E₁₆ or 56₈) literally
(case sensitive)
\d matches a digit (equivalent to [0-9])
> matches the character > with index 62₁₀ (3E₁₆ or 76₈) literally
(case sensitive)

▼ / <[^>]+> / gm
< matches the character < with index 60₁₀ (3C₁₆ or 74₈) literally
(case sensitive)
▼ Match a single character not present in the list below [>]
+ matches the previous token between one and unlimited
times, as many times as possible, giving back as needed
(greedy)
> matches the character > with index 62₁₀ (3E₁₆ or 76₈) literally
literally (case sensitive)
> matches the character > with index 62₁₀ (3E₁₆ or 76₈) literally
(case sensitive)
▼ Global pattern flags
g modifier: global. All matches (don't return after first
match)
m modifier: multi line. Causes ^ and \$ to match the
begin/end of each line (not only begin/end of string)