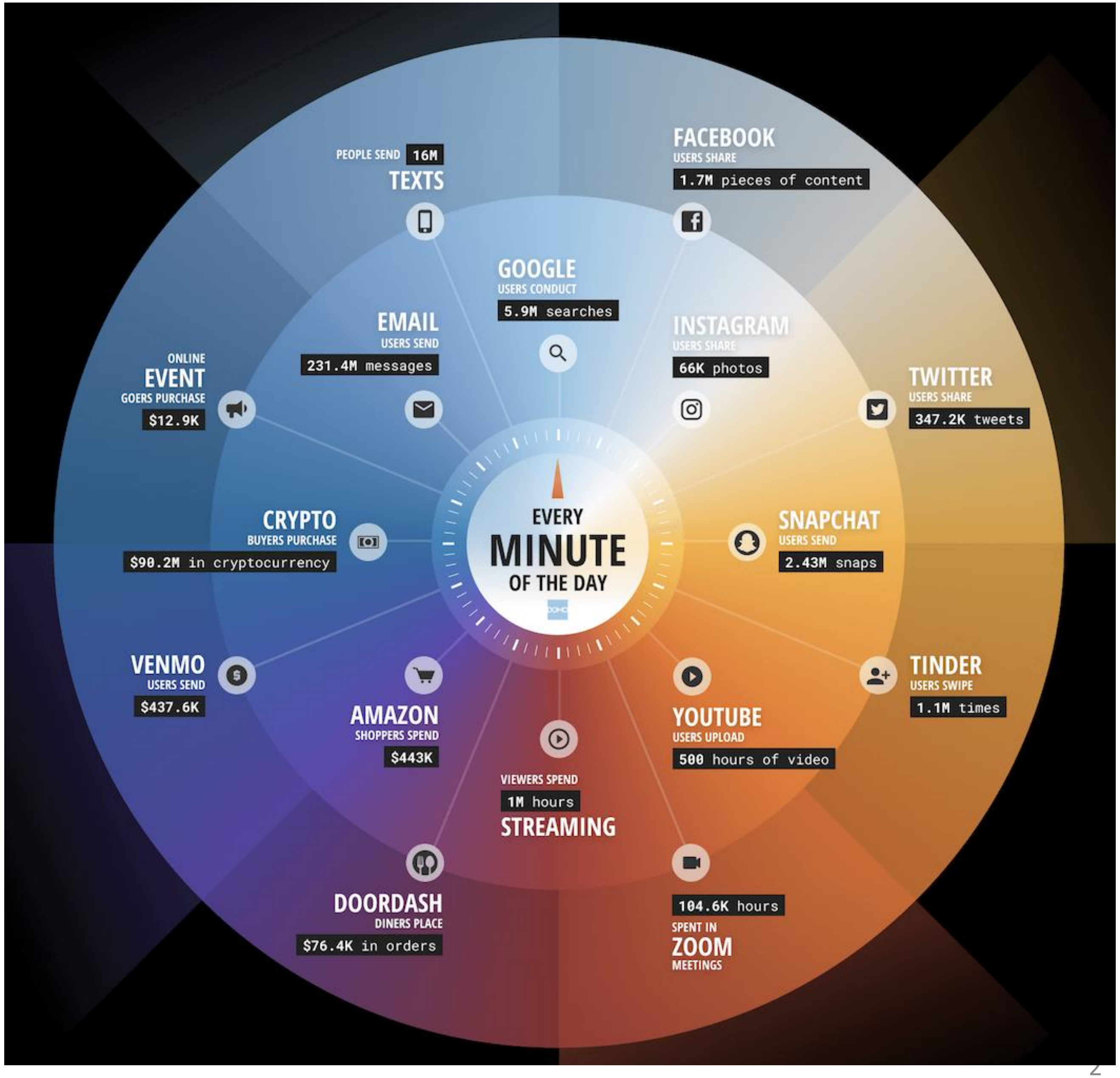
Nature of Text and Preprocessing

How much of this is textual data?

Explosion of Text data.



https://www.domo.com/data-never-sleeps

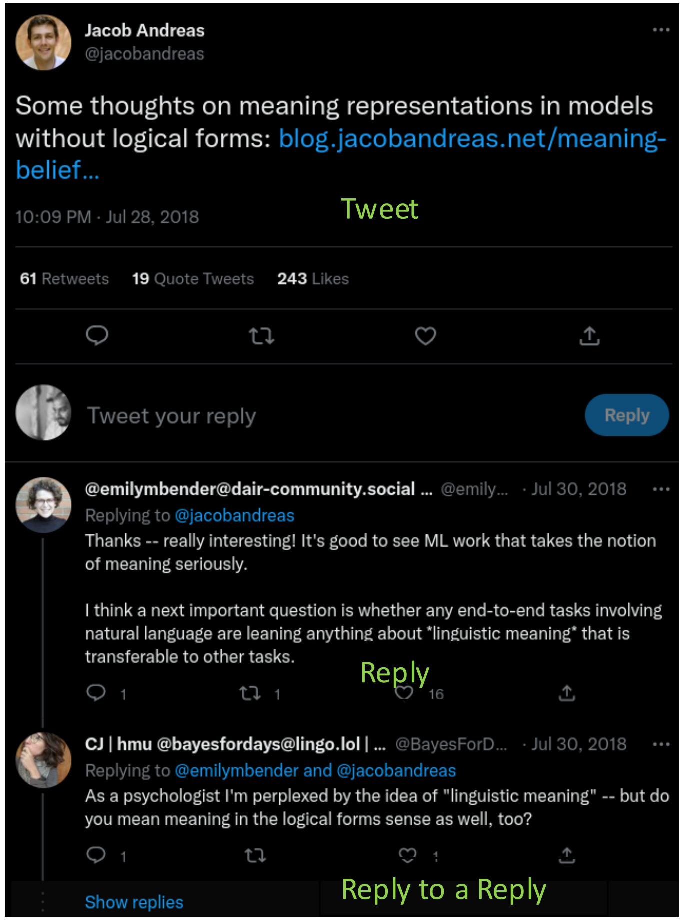
Where all can we find text in Online Social Networks

Social Network Posts – Tweets / Threads

Hierarchy in posts – Post
Reply
Reply-to-a-Reply

* and many such branches

Hierarchy encodes context that is crucial for processing text for meaningful tasks



[Jacob Andreas Tweet]

Where all can we find text in Online Social Networks

SideNote: This Twitter mega-thread, lead to an ACL'2020 Best Paper Award.

*ACL = Association of Computational Linguistics, a top-NLP conference.

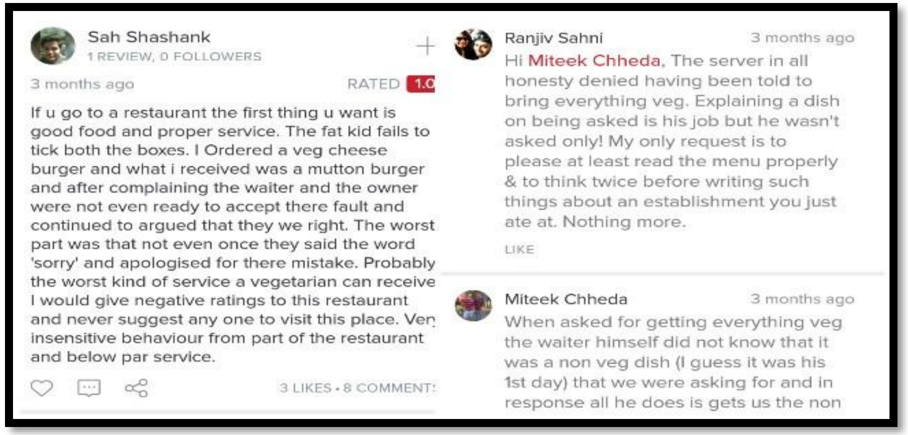
Acknowledgments. This paper benefitted from many inspiring and often spirited discussions. Without implying any agreement with the contents as presented, we thank Sam Bowman, Vera Demberg, Lucia Donatelli, Jason Eisner, Jonas Groschwitz, Kristen Howell, Angie McMillan-Major, Joakim Nivre, Stephan Oepen, Ellie Pavlick, Benjamin Roth, Dan Roth, Asad Sayeed, Hinrich Schütze, Nina Tahmasebi, and Olga Zamaraeva. This paper originated in a Twitter mega-thread that was neatly summarized by Thomas Wolf (2018). We also thank the ACL reviewers and the participants of the Toulouse Workshop on Formal and Distributional Semantics (2015) and *SEM 2016 for their insightful and constructive thoughts.

Where all can we find text in Online Social Networks

Product Reviews

https://www.cnet.com/pictures/funniest-amazon-reviews-products/10/

Restaurant Reviews



https://scroll.in/magazine/812253/x-why-indian-restaurants-are-yelling-back-at-negative-online-reviewers

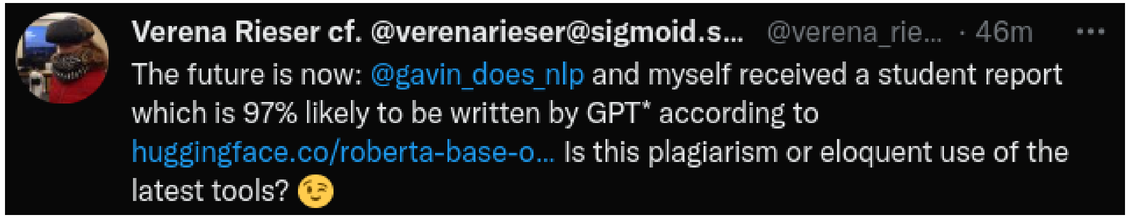
Artefacts of Social Media Text

Hashtags - #AI

Emojis - 😉

URLs : huggingface.co.....

User Mentions - @gavin_does_nlp



https://twitter.com/verena_rieser/status/1603305659117731841

Are these attributes even important to be kept as the part of the data?

Artefacts of Social Media Text

Hashtags
Emojis
URLs
User Mentions

Some carry semantic information - Hashtags

"#SuperMovie #Avengers" -> "Super Movie Avengers"

Others are useful in getting more information / context around a post - URLs

"Check out our model details at https......."

Quirks of social media text

Spelling Variations

Romanization : मेरा == mera , merra, meraa, mra

Character Limits : Good Night == Gud ni8

Romanization in any other language?

Quirks of social media text

Multimodal – Text makes sense when coupled with associated media.

https://twitter.com/introvertsmemes/status/1597203946774597636

On what platforms, multimodality is super important?

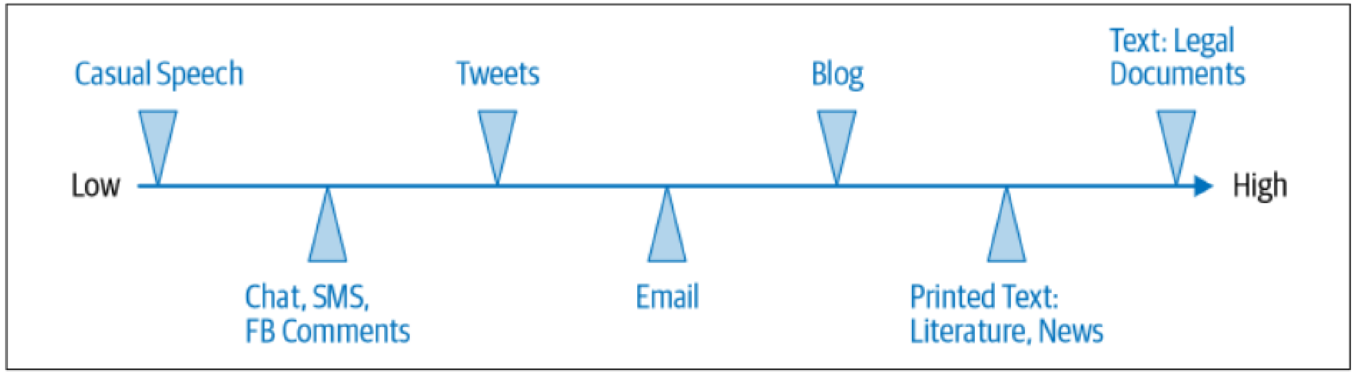
Quirks of social media text

Domain & Style Difference

Very different to, say, Wikipedia/News Text

Platform / User enforced Constraints : 280 Character Limit

All the information is not present in surface form – humor , sarcasm, anecdotes.



Spectrum of formalism in text data depending on data sources

Reference : [Practical Natural Language Processing by Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta, Harshit Surana]

Preprocessing

All the quirks that we discussed in previous slides have an impact on how we process social media text and gain insights

Text from Online Social Networks → Popular NLP Approaches/Models →

Preprocessing

Preprocessing Social Media Text usually has two objectives.

Preprocessing would help NLP tools perform better

Preprocessing social media text goes a long way in leveraging existing NLP toolkits

Mostly NLTK and Spacy libraries

Preprocessing

All the quirks that we discussed in previous slides have an impact on how we process social media text and gain insights

Most NLP toolkits are usually trained on News Corpus / Wikipedia Corpus. Because they are available on a large scale. But can these tools be used for social media text?

A lot of information can be extracted from artefacts. But they also cause noise

Preprocessing Social Media Text

- Preprocessing would help NLP tools perform better
- Preprocessing social media text goes a long way in leveraging existing NLP toolkits

Preprocessing

Cleaning Text – First step in any NLP Pipeline

Remove information that you don't want

OR

Refactor certain information in a certain way that helps your task

Preprocessing

Cleaning Text – Which output do you want?

I saw the new #johndoe movie
and it suuuuucks!!! WAISTED
$10... #badmovies :/"

Preprocessing

Cleaning Text – Which output do you want?

I saw the new #johndoe movie and it suuuuucks!!! WAISTED $10... #badmovies :/"

i saw the new <hashtag> john doe </hashtag> movie and it sucks <elongated> ! <repeated> waisted <allcaps> <money> . <repeated> <hashtag> bad movies </hashtag> <annoyed>

Replace artefacts with indicators.

Example taken from Ekpharasis's GitHub Example : https://github.com/cbaziotis/ekphrasis

Preprocessing

Cleaning Text – Which output do you want?

I saw the new #johndoe movie and it suuuuucks!!! WAISTED $10... #badmovies :/"

i saw the new <hashtag> john doe </hashtag> movie and it sucks <elongated> ! <repeated> waisted <allcaps> <money> . <repeated> <hashtag> bad movies </hashtag> <annoyed>

i saw the new john doe movie and it sucks ! waisted $10. bad movies

Which method is better?

Remove the artefacts completely.

Preprocessing

Cleaning Text – First step in any NLP pipeline

I saw the new #johndoe movie
and it suuuuucks!!! WAISTED
$10... #badmovies :/"

i saw the new <hashtag> john doe </hashtag>
movie and it sucks <elongated> !
<repeated> waisted <allcaps> <money> .
<repeated> <hashtag> bad movies
</hashtag> <annoyed>

i saw the new john doe movie and it sucks ! waisted
$10. bad movies

Replace / Remove – Depends on your application.

Preprocessing

Tokenization – Breaking a sentence into constituent tokens

I saw the new #johndoe movie and it suuuuucks!!! WAISTED $10... #badmovies >3:/

White Space

['I', 'saw', 'the', 'new', '#johndoe', 'movie', 'and', 'it', 'suuuuucks!!!', 'WAISTED', '$10...', '#badmovies', '>3:/']

Tool1

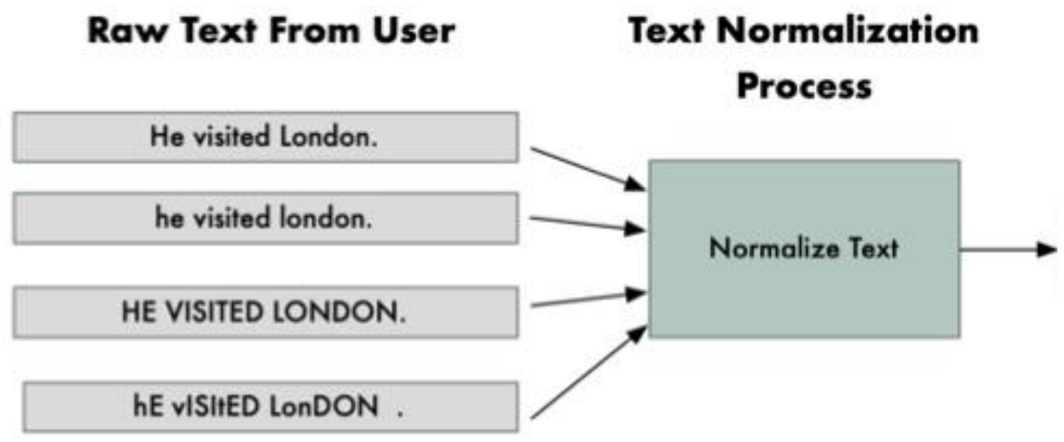
['I', 'saw', 'the', 'new', '#', 'johndoe', 'movie', 'and', 'it', 'suuuuucks', '!!!', 'WAISTED', '$', '10', '...', '#', 'badmovies', '>', '3', ':/']

Tool2

['I', 'saw', 'the', 'new', '#johndoe', 'movie', 'and', 'it', 'suuuuucks', '!', '!', '!', 'WAISTED', '$10', '.', '.', '.', '#badmovies', '>', '3:/']

Example taken from Ekpharasis's GitHub Example : https://github.com/cbaziotis/ekphrasis

Lower case



he visited london

Tokenization

- Input: Computer Science department at Virginia Tech
- Tokens: computer

  science

  department

  at

  virginia

  tech

Removing stop words

NLTK library

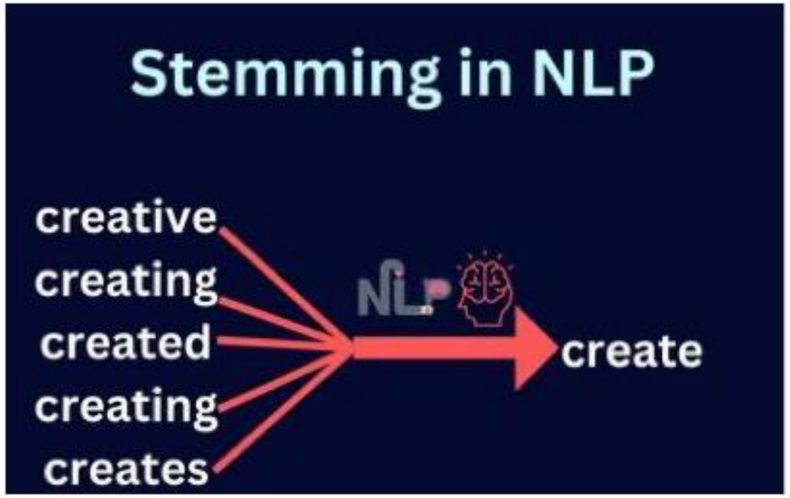
Output:

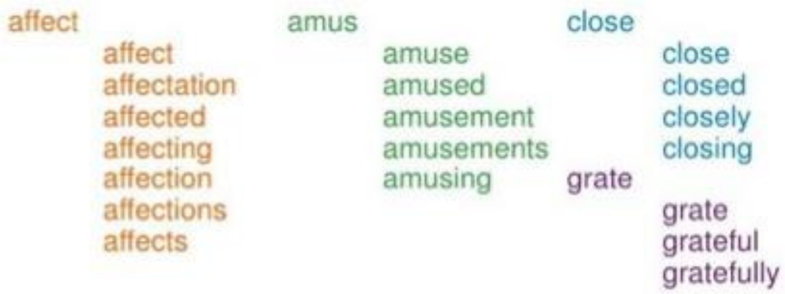
```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours',
'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself',
'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself',
'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are',
 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did',
'doing',
 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by',
'for',
'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above',
'below', 'to',
'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then',
'once',
'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',
'most',
'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm',
'o', 're', 've', 'y',
 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't",
'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
 "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't",
'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

Note: You can even modify the list by adding words of your choice in the English .txt. file in the stopwords directory.

Are there some useful tokens as well?

Stemming in NLP





The Porter Stemmer (Porter, 1980)

- Common Algorithm for English language
- A simple rule-based algorithm for stemming
- An example of a HEURISTIC method
- Based on rules like:
    - ATIONAL -> ATE (e.g., rel**ational** -> rel**ate**)
- The algorithm consists of 7 sets of rules, applied in order

tartarus.org/martin/PorterStemmer/def.txt

The Porter Stemmer: definitions

- Definitions:
  - **CONSONANTS**: a letter other than A, E, I, O, U, and Y preceded by consonant
  - **VOWEL**: any other letter (if the letter is not a consonant)
- With this definition, all words are of the form: $(C)(VC)^{m}(V)$

(C) And (V) are strings of consonants and vowels with 0 or more length, respectively

Measure of the word

- M=0 TREE, BY, TR

- M=1 TROUBLE, OATS, TREES, IVY

- M=2 TROUBLES, PRIVATE, OATEN

tartarus.org/martin/PorterStemmer/def.txt

The Porter Stemmer: Rule format

- The rules are of the form:
- (condition) S1 -> S2 where S1 and S2 are suffixes
- If the rule (m>1) EMENT->
  - In this S1 is EMENT and S2 is NULL
  - So, this would map REPLACEMENT with REPLAC

tartarus.org/martin/PorterStemmer/def.txt

Conditions

m	The measure of the stem
***S**	The stem ends with S
v	The stem contains a vowel
***d**	The stem ends with a double consonant (TT,SS)
***o**	The stem ends in CV C (second C not W, X, or Y) Ex: WIL, HOP

The condition may also contains expressions with and, or, or not
Example ((m>1) and (*s or*t)) -tests for a stem with m>1 ending in s or t

tartarus.org/martin/PorterStemmer/def.txt

The Porter Stemmer: Step 1

- SSES -> SS
  - caresses -> caress
- IES -> I
  - ponies -> poni
  - ties -> ti
- SS -> SS
  - caress -> caress
- S -> €
  - cats -> cat

tartarus.org/martin/PorterStemmer/def.txt

The Porter Stemmer: Step 2a (past tense, progressive)

- (m>0) EED -> EE
  - **Condition verified**: agreed -> agree
  - **Condition not verified**: feed -> feed
- ($*_v*$) ED -> €
  - **Condition verified**: plastered -> plaster
  - **Condition not verified**: bled -> bled
- ($*_v*$) ING -> €
  - **Condition verified:** motoring -> motor
  - **Condition not verified**: sing -> sing

tartarus.org/martin/PorterStemmer/def.txt

Why is “agreed” present in the first case but not in the second?

The Porter Stemmer: Step 2b (cleanup)

- (These rules are ran if second or third rule in 2a apply)
- AT -> ATE
  - Conflat(ed) -> conflate
- BL -> BLE
  - Troubl(ing) - > trouble
- (*d & ! (*L or *S or *Z)) -> single letter
  - **Condition verified**: hopp(ing) -> hop, tann(ed) -> tan
  - **Condition not verified**: fall(ing) -> fall
- (m=1 & *o) -> E
  - **Condition verified**: fil(ing) -> file
  - Condition not verified: fail -> fail

tartarus.org/martin/PorterStemmer/def.txt

The Porter Stemmer: step 3 and 4

- Step 3: Y elimination (*V*) Y -> I
  - **Condition verified**: happy -> happi
  - **Condition not verified**: sky -> sky
- Step 4: Derivational Morphology, I
  - (m>0) ATIONAL -> ATE
    - Relational -> relate
  - (m>0) IZATION -> IZE
    - Generalization -> generalize
  - (m>0) BILITI -> BLE
    - Sensibiliti -> sensible

tartarus.org/martin/PorterStemmer/def.txt

Porter Stemmer Step 5 and Step 6

- Derivational Morphology II
  - (m>0) ICATE-> IC
    - Triplicate-> Triplic
  - (m>0) FUL -> €
    - hopeful-> hope
  - (m>0) NESS-> €
    - goodness->good
- Derivational Morphology III
  - (m>1) ANCE-> €
    - allowance-> allow
  - (m>1) ENT -> €
    - dependent-> depend
  - (m>1) IVE-> €
    - effective->effect

tartarus.org/martin/PorterStemmer/def.txt

The porter stemmer Step 7 (cleanup)

- Step 7a
  - (m>1) E -> €
    - Probate -> probat
  - (m=1 & !*o) NESS -> €
    - Goodness -> good
- Step 7 b
  - (m>1 & *d & *L) -> single letter
    - **Condition verified**: controll -> control
    - **Condition not verified:** roll -> roll

tartarus.org/martin/PorterStemmer/def.txt

Lemmatization

- Task of determining whether two words have same root despite surface differences

Lemmatization

- Dictionary based technique to convert into root form
- “better” is converted to “good”
- “is, are, am” to “be”
- POS tags for context
- POS tags differentiating between “I saw a book” VS “I cut the material using a saw”

Comparison of stemming and lemmatization?

Stemming vs Lemmatization

Stemming

achieve -> achiev
achieving -> achiev

- Can reduce words to a stem that is not an existing word
- Operates on a single word without knowledge of the context
- Simpler and faster

Lemmatization

achieve -> achieve
achieving -> achieve

- Reduces inflected words to their lemma, which is always an existing word
- Can leverage context to find the correct lemma of a word
- More accurate but slower

43

Acknowledgements

- Dr. Ponnurangam Kumaraguru at IIIT Hyderabad
- Dr. Nidhi Goel at Mahindra University