13 – Automatic Language Correction

IA161 Advanced Techniques of Natural Language Processing

A. Horák, J. Švec

NLP Centre, FI MU, Brno

January 14, 2021

Motivation

This tool can be use to find spelling , gramar or stylistic errors in english texts. just paste some text in the the box and click 'Submit to check . Additionally, their are many different dialects you can chose from. Additionally , you can hover your mouse over a error to see it's description and an useful list of posible corrections. You don´t need to worry for your writing skills any more, improving you're text has never be more easier!

Types of errors[1]:

Grammar (6) | Spelling (10) | Other (2) | Spacing (3) | Typographical (2) | Duplication (1)

[1] Source: `http://www.onlinecorrection.com/`

Automatic language correction

A text with errors…

- is less comprehensible,
- looks less professional,
- poses problems for machine translation

People are quite resilient to letter-switching errors:

Example (Cmabrigde Uinervtisy (Cambridge University) effect)

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

Example by Davis, M. 2003. Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy
`http://www.mrc-cbu.cam.ac.uk/people/matt.davis/cmabridge/`

Automatic language correction

Automatic language correction:

- spell checking – detect spelling errors in individual words,
- grammar checking – incorrect use of person, number, case or gender, improper verb government, wrong word order, etc. . .
- word completion – suggestion of the word currently being entered.

Spell checking

- detecting which words in a document are misspelled,
- providing spelling suggestions for incorrectly spelled words in a text,
- correction is the task of substituting the well-spelled hypotheses for misspellings,
- usually uses a dictionary of valid words,
- application: word processing and postprocessing optical character recognition [Whitelaw et al., 2009] or speech recognition.

Type of errors

- Non-word errors – the misspelled word is not a valid word in a language,
  - typographic errors – usually keyboard typing error (e.g. "teh" – "the", "speel" – "spell"),
  - cognitive errors – caused by the writer's misconceptions (e.g. "recieve" – "receive", "conspiricy" – "conspiracy"),
  - phonetic errors – substituting a phonetically equivalent sequence of letters (e.g. "seperate" – "separate").
- Real-word errors – sentence contains a valid word, but it is inappropriate in the context [Hladek et al., 2013].

Example

Non-word error: "I'd like a peice of cake."
Real-word error: "I'd like a peace of cake."

Error correction

- Consists of two steps:
  - generation of candidate corrections,
  - ranking of candidate corrections.
- Isolated-word methods:
  - edit distance,
  - similarity keys,
  - character n-gram-based techniques,
  - rule-based techniques,
  - probabilistic techniques,
  - neural networks [Sakaguchi et al., 2017].

Isolated-word methods I

Edit distance

- assumption – person usually makes few errors,
- minimum set of operations to transform a non-word to a dictionary word,
- operations: insertions, deletions and substitutions,
- useful for: correcting errors resulting from keyboard input.

Example

Edit distance between "kitten" and "sitting" is 3:

1. kitten $\rightarrow$ sitten   substitution of "s" for "k"
2. sitten $\rightarrow$ sittin   substitution of "i" for "e"
3. sittin $\rightarrow$ sitting   insertion of "g" at the end

Isolated-word methods II

Similarity keys:

- assign a key to each dictionary word,
- compare with the key computed for the non word,
- most similar key is selected as suggestion.

Soundex – phonetic algorithm (English) [Holmes and McCabe, 2002]

Example

N	Represents letters
1	B, F, P, V
2	C, G, J, K, Q, S, X, Z
3	D, T
4	L
5	M, N
6	R

1. Keep the first letter
2. Drop occurrences of `a`, `e`, `i`, `o`, `u`, `y`, `h`, `w`
3. Replace letters with numbers
4. Merge adjacent identical numbers
5. Add zeroes to the end, or remove right-most numbers

Output: `(letter, number, number, number)`

key("Robert")=R163; key("Robin")=R150 – not similar
key("Smith")=S530; key("Smyth")=S530 – similar

Isolated-word methods III

Character N-gram-based techniques:

- compute similarity coefficient of two strings
- based on the number of shared n-grams (*Jaccard similarity*)

$$\delta_n(a,b) = \frac{|\textit{n-grams}(a) \cap \textit{n-grams}(b)|}{|\textit{n-grams}(a) \cup \textit{n-grams}(b)|}$$

Example

fact vs. fract

$$\begin{aligned}
\textit{bigrams}(\text{“fact”}) &= \{\text{“-f”}, \text{“fa”}, \text{“ac”}, \text{“ct”}, \text{“t-”}\} && \text{... 5 bigrams}\\
\textit{bigrams}(\text{“fract”}) &= \{\text{“-f”}, \text{“fr”}, \text{“ra”}, \text{“ac”}, \text{“ct”}, \text{“t-”}\} && \text{... 6 bigrams}\\
\ldots \cap \ldots &= \{\text{“-f”}, \text{“ac”}, \text{“ct”}, \text{“t-”}\} && \text{... 4 bigrams}\\
\ldots \cup \ldots &= \{\text{“-f”}, \text{“fa”}, \text{“fr”}, \text{“ra”}, \text{“ac”}, \text{“ct”}, \text{“t-”}\} && \text{... 7 bigrams}
\end{aligned}$$

$$\delta_2(\text{“fact”}, \text{“fract”}) = \tfrac{4}{7} = 0.57$$

Isolated-word methods IV

Rule-based techniques

- a set of rules for common misspellings and typographic errors,
- each rule "fixes" one kind of error
- rules are applied to out-of-vocabulary words

Probabilistic techniques

- based on statistical features of the language (corpus)
  - transition probabilities – probability that a letter is followed by another letter
  - confusion probabilities – how often a letter is mistaken or substituted for another letter

Neural networks

- employs neural language models for context
- word-based – input node = every possible n-gram in every position of a word
- output node for each word in the dictionary
- character-based with recurrent neural networks

Outline

Grammar checking

Example

"That's good to now"
"That's good to know"

Grammar checking starts where spell checking ends

- deals with the most difficult and complex type of language errors
  - wrong word order,
  - verb tense errors,
  - subject/verb agreement,
  - punctuation errors,
  - etc...
- two main approaches
  - rule-based methods – time-consuming, less flexible, more precise better interpretability
  - statistical methods – easier and faster to implement, learn from examples
    need a lot of data [Nazar and Renau, 2012]

Rule-based grammar checking

Testing the input text against a set of handcrafted rules

Example

rule: I + verb(3rd person, singular form)
→ incorrect verb form usage – "I has a dog"

- ➕ advantages:
  - rules can be easily added, modified or removed
  - rule can have a corresponding extensive explanation,
  - decisions can be traced to a particular rule,
  - rules can be authored by linguists, no need of programming
- ➖ disadvantages:
  - large amount of manual work
  - extensive rule set is needed [Mozgovoy, 2011].

Rule-based grammar checker example

LanguageTool[2] – open source grammar checker

1. plain text as input
2. splits text into sentences
3. splits sentences into words
4. finds part-of-speech tags for each word and its base form
   walks – walk
5. matches the analyzed sentences against error patterns and runs rules.

[2] `https://languagetool.org/` [Naber, 2003, Brenneis, 2018]

Rule example in LanguageTool

Example

"I thing that's a good idea."

```
<rule id="YOU_THING" name="Possible typo 'I/you/... thing(think)'">

<pattern mark_from="1">
        <token regexp="yes">I|you</token>
        <token regexp="yes">thing|things</token>
</pattern>

<message>Did you mean <suggestion>think</suggestion> ?</message>
<example type="correct">I <marker>think</marker> that's a good idea.</example>

</rule>
```

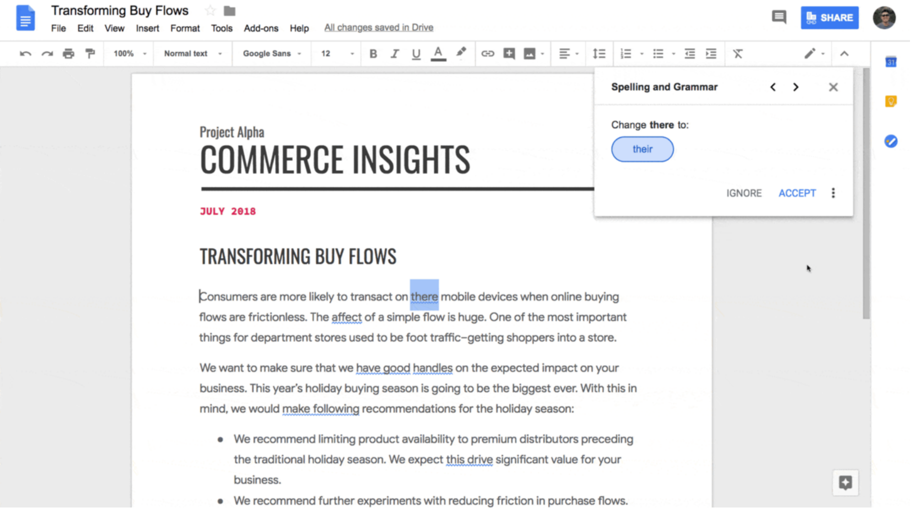
Statistical grammar checking

- based on analysis of grammatically correct POS-annotated corpus,
- build a list of POS tag sequences,
  - some sequences are very common (determiner+adjective+noun as in "the old man")
  - others will probably not occur at all (determiner+determiner+adjective)
- sequences which occur often in the corpus are considered correct,
- uncommon sequences might be errors.

Google Grammar Checker

- available in Google Docs since 2019
- based on neural machine translation architecture
- trains to translate incorrect language → correct language [Grundkiewicz and Junczys-Dowmunt, 2018]

Google Grammar Checker



Outline

Word completion

- reduce the number of keystrokes
- suggesting the completion of the word
- use context information to predict what block of characters (letters, n-grams, syllables, words, or entire phrases) a person is going to write next
- based on wide-coverage word or language model
- prediction at earliest possible point of a character sequence being entered [Van den Bosch, 2011]

Best results

- Spell checking (first suggestion):
  - English – 97 % [Sakaguchi et al., 2017]
  - Czech – 75 % [Ramasamy et al., 2015, Richter et al., 2012]
- Grammar checking (various tests average):
  - English – 72 % [Grundkiewicz and Junczys-Dowmunt, 2018]
  - Czech – 40 % [Petkevič, 2014]

References I

Brenneis, M. (2018).
Development of neural network based rules for confusion set disambiguation in languagetool.
SKILL 2018-Studierendenkonferenz Informatik.

Grundkiewicz, R. and Junczys-Dowmunt, M. (2018).
Near human-level performance in grammatical error correction with hybrid machine translation.
arXiv preprint arXiv:1804.05945.

Hladek, D., Stas, J., and Juhar, J. (2013).
Unsupervised spelling correction for Slovak.
Advances in Electrical and Electronic Engineering, 11(5):392–397.

Holmes, D. and McCabe, M. C. (2002).
Improving precision and recall for soundex retrieval.
In *Information Technology: Coding and Computing, 2002. Proceedings. International Conference on*, pages 22–26. IEEE.

References II

Mozgovoy, M. (2011).
Dependency-based rules for grammar checking with LanguageTool.
In *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*, pages 209–212.

Naber, D. (2003).
A rule-based style and grammar checker.

Nazar, R. and Renau, I. (2012).
Google books n-gram corpus used as a grammar checker.
In *Proceedings of the Second Workshop on Computational Linguistics and Writing (CLW 2012): Linguistic and Cognitive Aspects of Document Creation and Document Engineering*, EACL 2012, pages 27–34, Stroudsburg, PA, USA. Association for Computational Linguistics.

References III

Petkevič, V. (2014).
Kontrola české gramatiky (český grammar checker).
Studie z aplikované lingvistiky - Studies in Applied Linguistics, 5(2):48–66.

Ramasamy, L., Rosen, A., and Stranák, P. (2015).
Improvements to korektor: A case study with native and non-native czech.
In *ITAT*, pages 73–80.

Richter, M., Straňák, P., and Rosen, A. (2012).
Korektor-a system for contextual spell-checking and diacritics completion.
In *COLING (Posters)*, pages 1019–1028.

References IV

Sakaguchi, K., Duh, K., Post, M., and Van Durme, B. (2017).
Robsut wrod reocginiton via semi-character recurrent neural network.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Van den Bosch, A. (2011).
Effects of context and recency in scaled word completion.
Computational Linguistics in the Netherlands Journal, 1.

Whitelaw, C., Hutchinson, B., Chung, G. Y., and Ellis, G. (2009).
Using the web for language independent spellchecking and autocorrection.
In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 890–899, Stroudsburg, PA, USA. Association for Computational Linguistics.