Homophone Auto-Checker with Web App

Austin Barish, Marion Bauman, Hongxin Wu

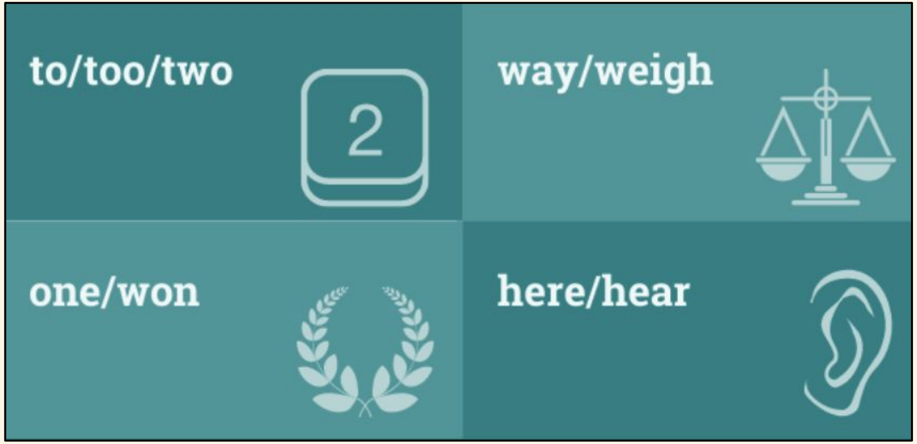
Table of contents

01

Introduction

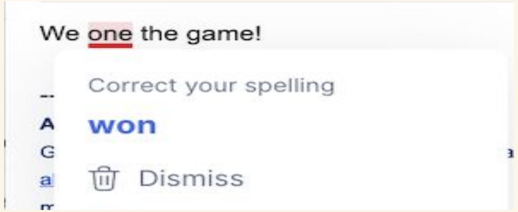
What is a homophone?

- “one of two or more words pronounced alike but different in meaning or derivation or spelling (such as the words to, too, and two)”.
- Varies by dialect:
  - Entirely dependent on how you pronounce words.
  - A homophone for an Australian might not be for an American.
  - We focus on primarily American English homophones.

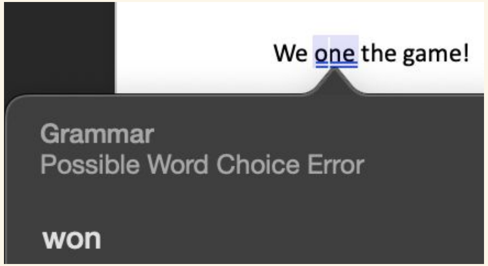


Existing Homophone Checkers

Grammarly



Word



Google Docs

iMessage

N/A

Goals

- Create a sufficient large test dataset to evaluate the model's accuracy, weak points, and best language model.
- Create a model capable of correctly identifying incorrectly used homophones and replacing them.
- Develop a web application to enable us, and anyone else, to test the model.
- Understand the challenges of correcting homophone mistakes and why some grammar checks often miss them.

02

Model and Test Data

Test Data Creation

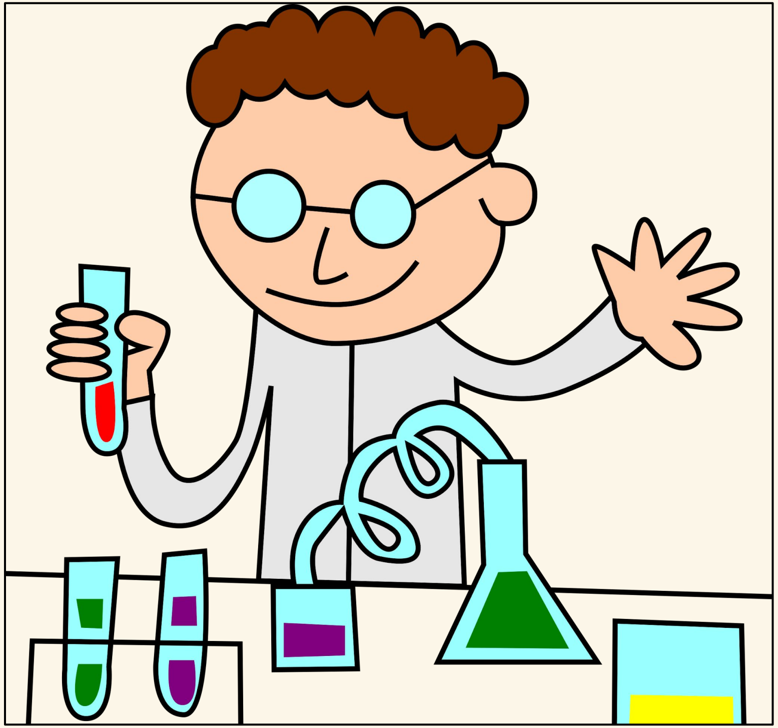
Create a sufficient large test dataset to evaluate the model’s accuracy, weak points, and best language model.

Importing Grammatically Correct Corpus

- Assuming that published text, should not have any homophone mistakes
- Imported the top 10* most downloaded books on Project Gutenberg
- Very simple text cleaning and sentence detection using NLTK's 'punkt' tokenizer.
- 68,573 total sentences.

Mistake Insertion

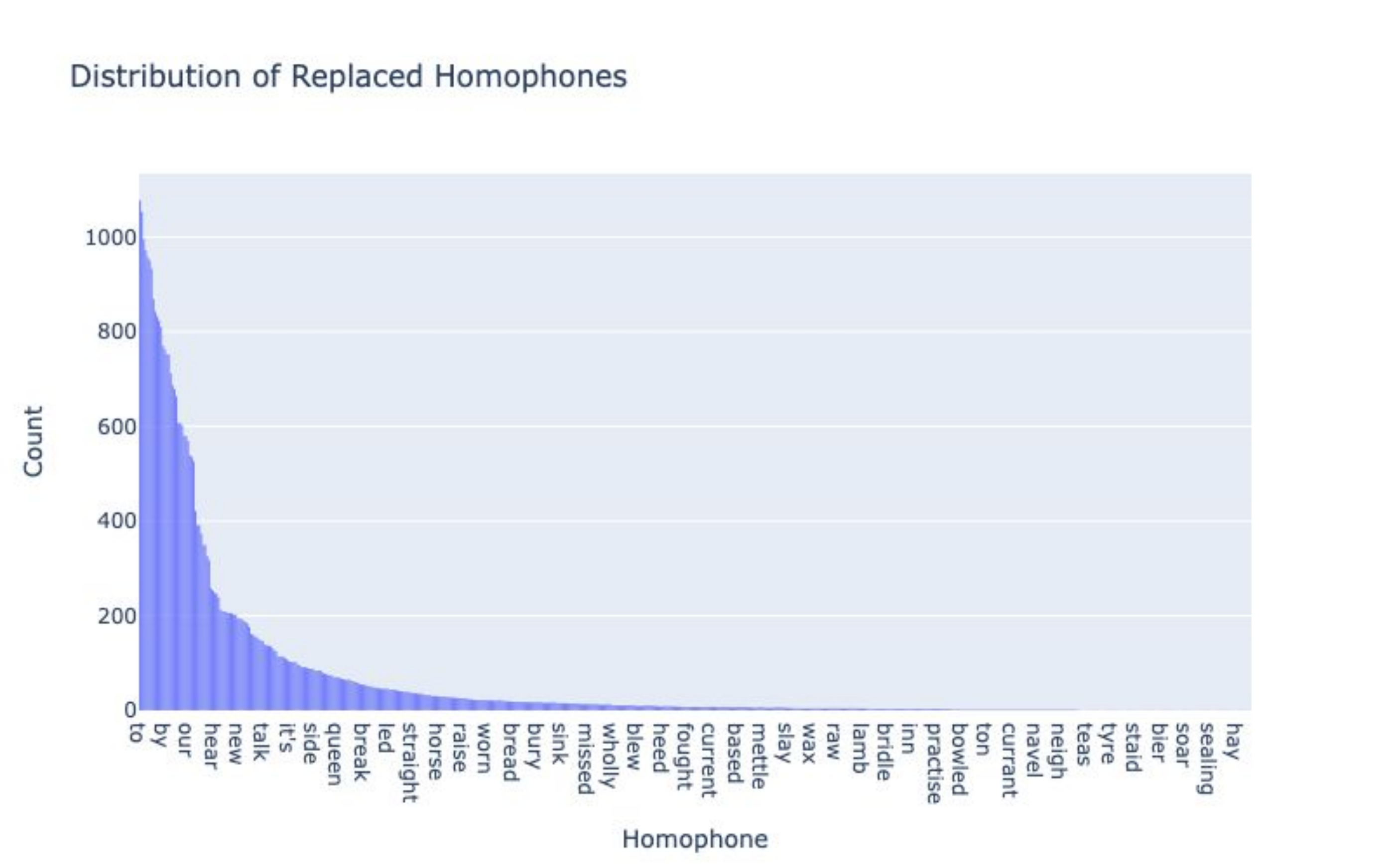
- Created a list of 442 sets of homophones ([to, too, two]) with 941 total homophones.
- Inserted a mistaken homophone into a sentence with probability p=0.7.
- Weighted the homophones to replace in order to create a more balanced distribution of wrongfully used homophones.



Homophone Replacement Probability Weights

- `count` = total appearances of the homophone in the current dataset
- `max_count` = maximum count of homophones in the sentence

$$w_i = 1 - \frac{\text{count}}{\text{max\_count} + \epsilon}, \epsilon = 1e - 10$$


Distribution of Replaced Homophones
Count
1000
800
600
400
200
0
to
by
our
hear
new
talk
it's
side
queen
break
led
straight
horse
raise
worn
bread
bury
sink
missed
wholly
blew
heed
fought
current
based
mettle
slay
wax
raw
lamb
bridle
inn
practise
bowled
ton
currant
navel
neigh
teas
tyre
staid
bier
soar
sealing
hay
Homophone

Final Test Data

	sentence	has_homophone	is_error	error_idx	error	correct_word	correct_sentence
0	the project gutenberg ebook of frankenstein; ...	True	False	NaN	NaN	NaN	the project gutenberg ebook of frankenstein; ...
1	you may copy it, give it aweigh or re-use it u...	True	True	6.0	aweigh	away	you may copy it, give it away or re-use it und...
2	if you are not located in the united states,yo...	True	True	11.0	two	to	if you are not located in the united states,yo...
3	html version by al haines.	True	False	NaN	NaN	NaN	html version by al haines.
4	further corrections buy menno de leeuw.	True	True	2.0	buy	by	further corrections by menno de leeuw.

- The final data frame has a shape of (68573, 7)
- 56,484 (82.37%) of these sentences contain at least one homophone.
- There are 39,446 (57.53%) sentences containing homophone errors.
- The most commonly replaced homophones were “to”, “in”, “you”, “for”, and “but”.

68573 Rows **7** Columns

Final data frame shape

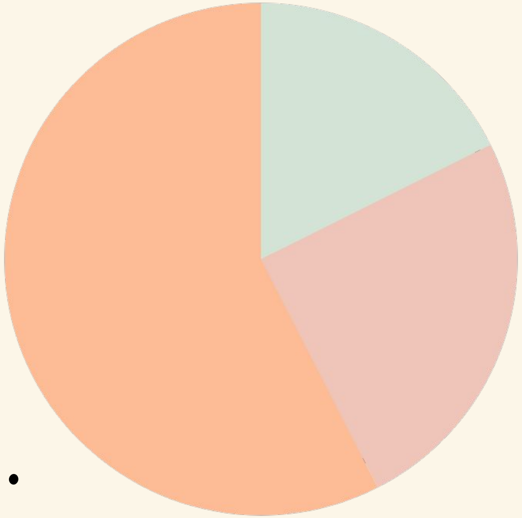
56,484

Sentences contain at least one homophone

39,446

Sentences containing homophone errors

Test Data Distribution



57.52% — **Homophone WITH Error**
sentences containing homophone errors

24.85% — **Homophone NO Error**
sentences containing homophone

17.63% — **NO homophone**
sentences containing no homophone

Homophone Checking Model

Model Process

1. Check for Homophone(s)

Uses the same homophone list as before

2. Masks Homophone(s)

We [MASK] the game!

3. Deploys BERT

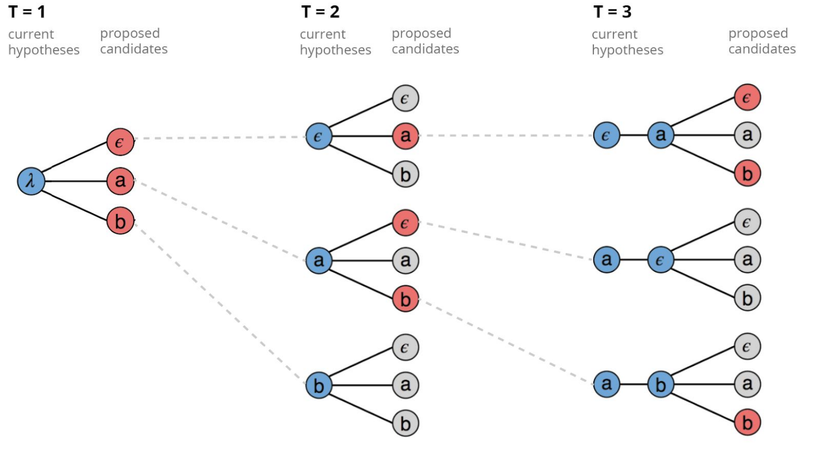
Checks for 50 most likely tokens at mask location

4. Return Sentence

Either return a similar data frame or just the final sentence

Homophone Masking using Beam Search

- Unilaterally masks the homophones moving from left to write.
    - Each homophone is tested on its own
- If a previous homophone was changed, the new masked string is the most correct version of the sentence
- Ex: “I eight two much food!”
    1. I [MASK] two much food!
    2. I ate [MASK] much food!



Standard beam search algorithm with an output alphabet {ε,a,b} and a beam size of three.

BERT
Fill-Mask

- Uses BERT Base Model through Hugging Face’s fill-mask tool
- Returns the top 50 most likely tokens and their probabilities
    - Need to return a high quantity of tokens as some homophone locations leave room for numerous options.
- Replace the homophone if one of its counterparts is more likely

03

Results

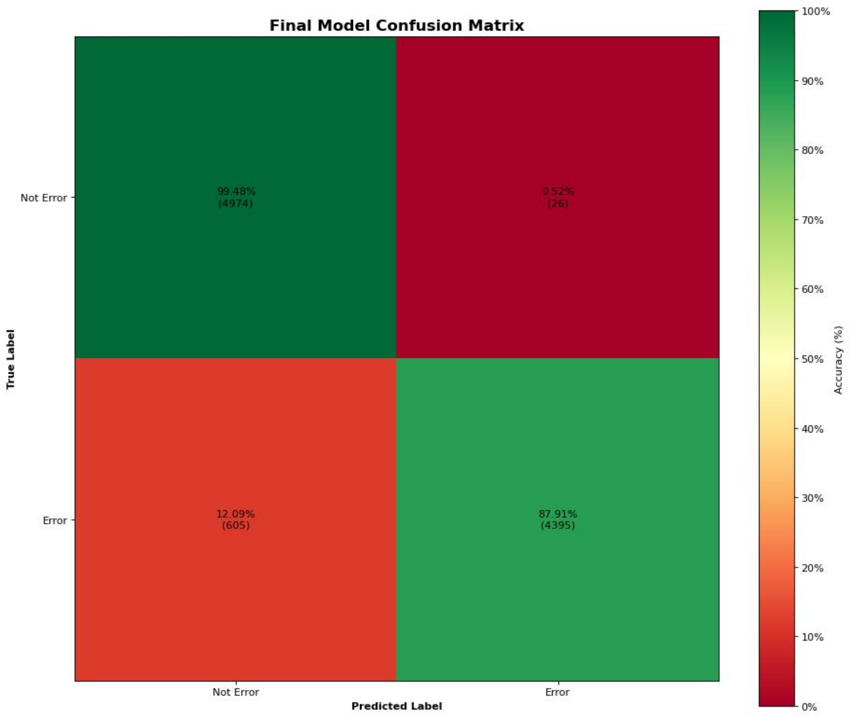
Model Testing

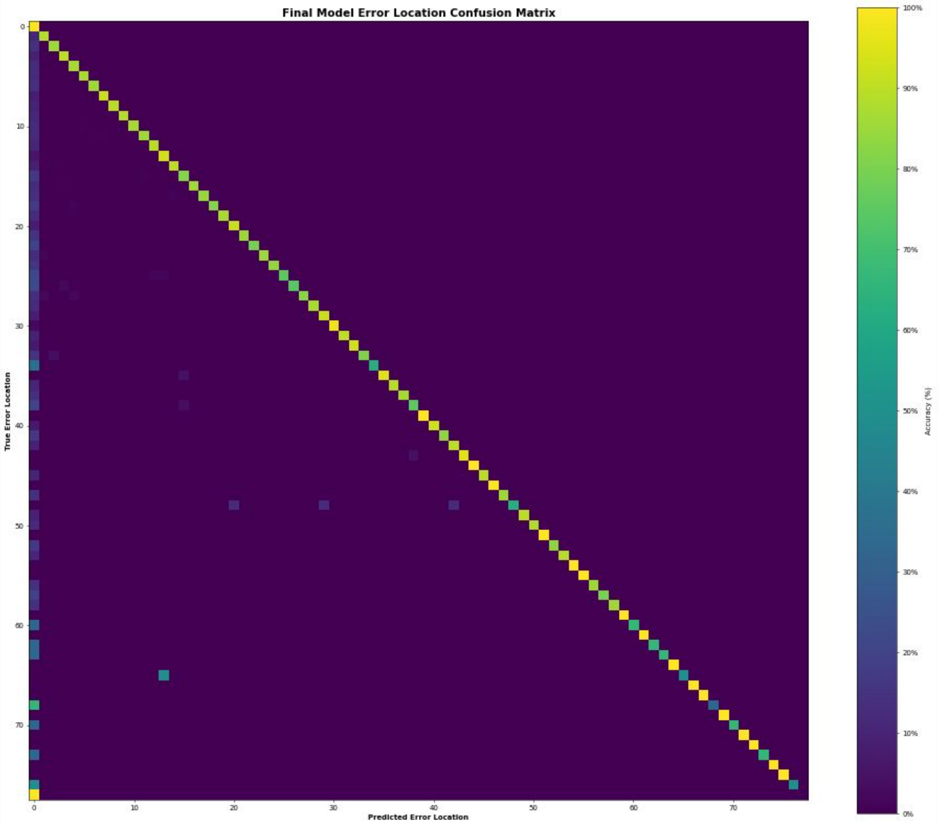
- Goal is to determine if model is **detecting errors**
  - Our model is great at fixing errors, but we are reliant on a language model to detect them
- Metrics:
  - Accuracy
  - Precision
  - Recall
  - Speed

Model Comparison

Model	Accuracy	Precision (Error)	Precision (No Error)	Test Size
bert-base-uncased	**0.928**	**0.996**	0.859	10,000
XLM–MLM-EN-2048	0.728	0.600	0.745	250
roberta-base	0.806	0.970	0.781	500
albert-base-v2	0.910	0.983	**0.888**	500

- Our model performs very well with high accuracy and low latency
- Very low false positive rate (0.52%) means that our model won't make unnecessary changes to text





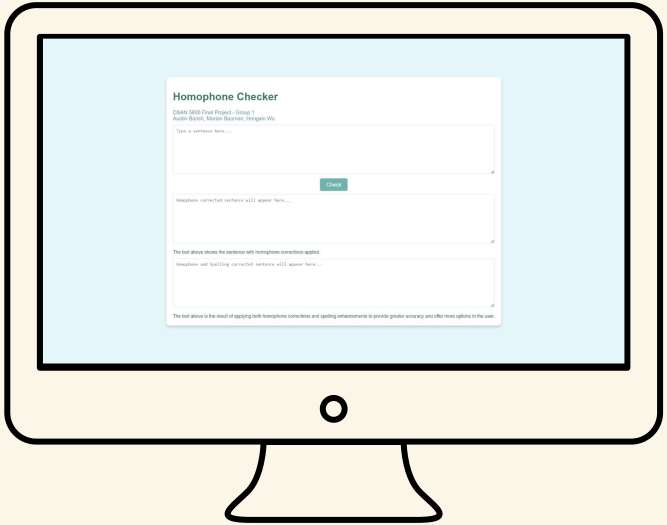
- Model performs equally well for sentences with multiple homophones or homophones in different locations
- Used -1 to indicate when the model predict no errors
- Worst performance when the error is at index 33

04

Web Application

Web App

See our demonstration!



How We Built It?

app.py

Handling all web server operations, routing HTTP requests, integrating homophone and spelling correction logic

homophone_utils

Identify and correct homophones in text, further enhanced by adding a spelling correction feature

index.html

Input text, initiate the correction process, and receive immediate updated feedback directly in web browser

app.py - Orchestrating the Web Application

Serves as the central file for our Flask-based web server, establishing the foundation of our web application

Main Flask Server File

Manages HTTP requests and responses, orchestrating the seamless flow of data between user interface and backend logic

Handling HTTP Interactions

Integrates homophone correction logic from homophone_utils.py and sends the processed data to the frontend for display

Integration with Homophone Correction

homophone_utils.py - Text Processing Engine

Implements logic to detect and rectify homophones in user-input sentences, ensuring textual accuracy and coherence

Homophone Identification and Correction

Leverages natural language processing techniques and the BERT (bert-base-uncased) model for context-aware homophone corrections

Utilizing NLP and BERT Model

Enhances the text correction capabilities by integrating the oliverguhr/spelling-correction-english-base model for additional spelling accuracy

Incorporating Spelling Correction

index.html - Interactive User Interface

Designed with a focus on simplicity and user-friendliness, enabling effortless interaction and engagement with the web application

Ease of Interaction

Provides two distinct output options: one showing corrections for homophones and another for combined homophone and spelling corrections, allowing users to compare and evaluate

Dual Output for Comparative Analysis

Employs AJAX technology for seamless data transmission between the frontend and backend, ensuring real-time updates without the need to refresh the page

Leveraging AJAX for Dynamic Interaction

Deployment

Utilize Hugging Face's free API to get model outputs without needing to download them locally

Hugging Face Inference Endpoints

Deploy the Flask application to Heroku, a cloud service for hosting web apps owned by Salesforce

Heroku Server

Create a report website using Quarto and link our Heroku app

Quarto Website

05

Conclusions

Conclusions

Key Achievements

- Developed a BERT-based model effectively correcting homophones, filling a crucial gap in conventional grammar-check tools.
- Generated a substantial dataset from Project Gutenberg and tested the model's robustness across 68,573 sentences. The model achieved a 92.8% accuracy rate in detecting and correcting homophones, and a 99.6% precision rate in correcting detected error homophones over 10,000 testing sentences.
- Learned the difficulties of correcting homophone mistakes and can recognize why many public models struggle with them or choose to ignore them.

Web Application Impact

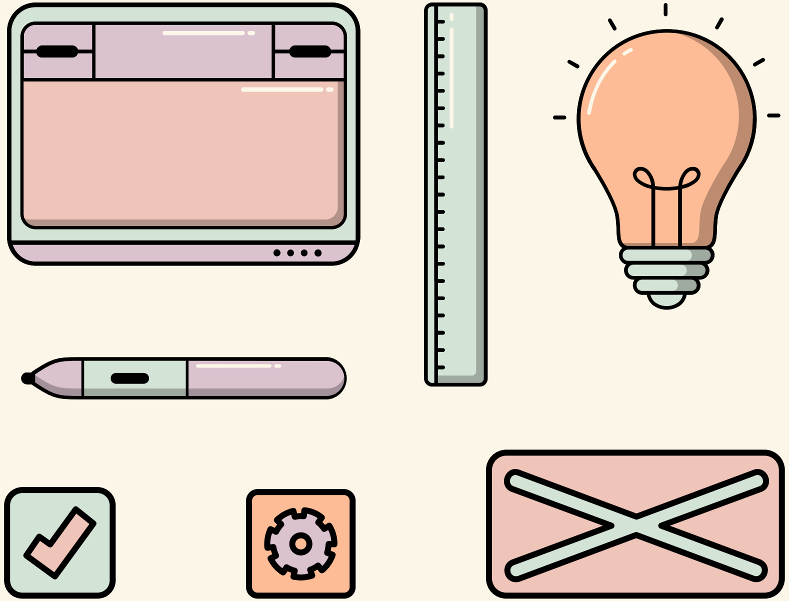
- Translated the model into a user-friendly web app, offering real-time corrections with dual output options for homophones and combined spelling errors.

Future Directions

- Potential model enhancements for greater accuracy and expanding the homophone list to cover more dialects and colloquialisms.

References

- Bert-base-cased
- Xlm-mlm-en-2048
- Albert-base-v2
- Roberta-base
- oliverguhr/spelling-correction-english-base



Thanks!

Does anyone have any questions?