

SPELLING CORRECTOR

PROBLEM STATEMENT

In this project we will be using deep learning approaches to build a spelling corrector.

DESCRIPTION OVERVIEW

A word needs to be checked for spelling correctness and corrected if necessary, many a time in the *context* of the surrounding words. A spellchecker points to spelling errors and possibly suggests alternatives. An autocorrector usually goes a step further and automatically picks the most likely word. In case of the correct word already having been typed, the same is retained.

There are different types of spelling errors.

1. **Non-word Errors:** These are the most common type of errors. You either miss a few keystrokes or let your fingers hurtle a bit longer. E.g., typing langage when you meant language; or hurryu when you meant hurry
2. **Real Word Errors:** If you have fat fingers, sometimes instead of creating a non-word, you end up creating a real word, but one you didn't intend. E.g, typing buckled when you meant bucked. Or your fingers are a tad wonky, and you type in three when you meant there.
3. **Cognitive Errors:** The previous two types of errors result not from ignorance of a word or its correct spelling. Cognitive errors can occur due to those factors. The words piece and peace are homophones (sound the same). So you are not sure which one is which. Sometimes your damn sure about your spellings despite a few grammar nazis claim you're not.
4. **Short forms/Slang/Lingo:** These are possibly not even spelling errors. May be u r just being kewl. Or you are trying hard to fit in everything within a text message or a tweet and must commit a spelling sin. We mention them here for the sake of completeness.

TECHNOLOGY USE

Here we will be using Anaconda Python 3.6 , Keras 2.2.4 using TensorFlow GPU 1.14.0 backend CUDA 10 with CuDNN 10

INSTALLATION

Installation of this project is pretty easy. Please do follow the following steps to create a virtual environment and then install the necessary packages in the following environment.

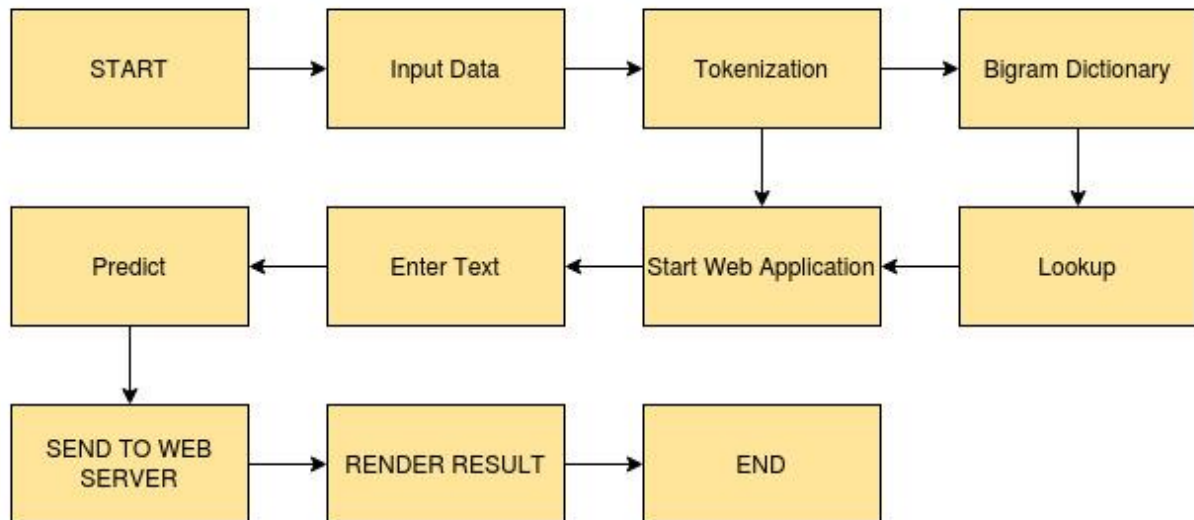
In Pycharm it's easy

1. Create a new project.
2. Navigate to the directory of the project
3. Select the option to create a new new virtual environment using conda with python3.6
4. Finally create the project using used resources.
5. After the project has been created, install the necessary packages from requirements.txt file using the command `pip install -r requirements.txt`

In Conda also it's easy

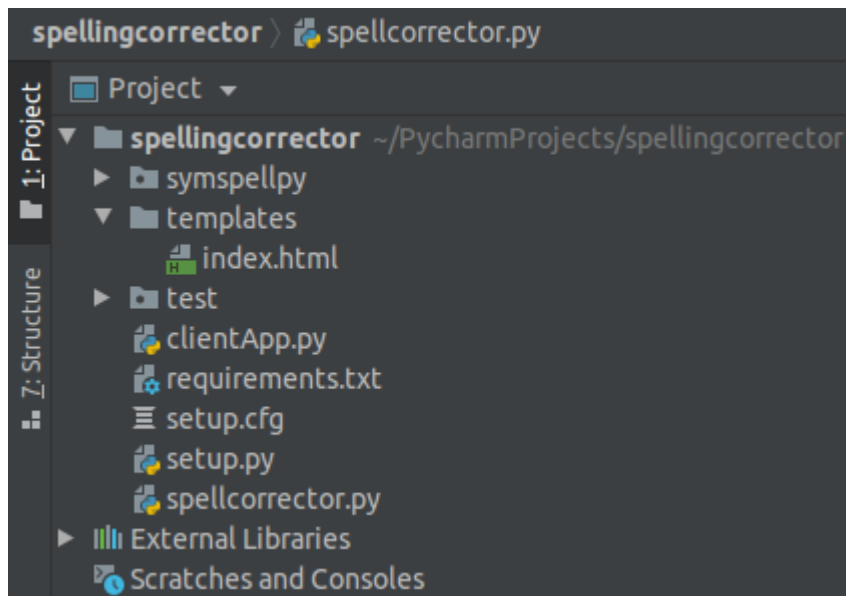
1. Create a new virtual environment using the command
`conda create -n your_env_name python=3.6`
2. Navigate to the project directory.
3. Install the necessary packages from requirements.txt file using the command
`pip install -r requirements.txt`

WORKFLOW DIAGRAM



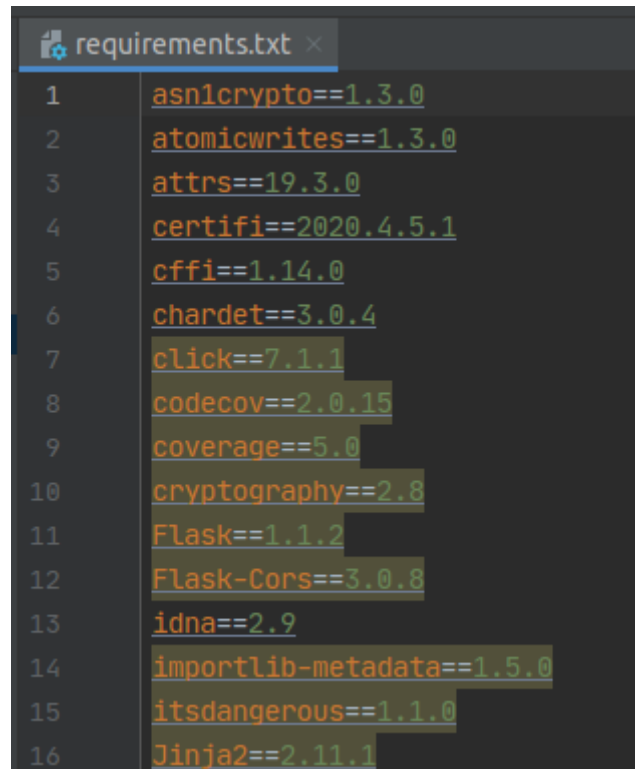
IMPLEMENTATION

1. Project Directory



This above picture is of the project directory if we open the project folder using Pycharm. In the directory different types of files are there like for spellingcorrector, flask server etc.

2. requirements.txt

A screenshot of a code editor showing a file named 'requirements.txt'. The file contains 16 lines of text, each representing a package and its version. The lines are numbered 1 through 16 on the left side. The packages and versions are: 1. asn1crypto==1.3.0, 2. atomicwrites==1.3.0, 3. attrs==19.3.0, 4. certifi==2020.4.5.1, 5. cffi==1.14.0, 6. chardet==3.0.4, 7. click==7.1.1, 8. codecov==2.0.15, 9. coverage==5.0, 10. cryptography==2.8, 11. Flask==1.1.2, 12. Flask-Cors==3.0.8, 13. idna==2.9, 14. importlib-metadata==1.5.0, 15. itsdangerous==1.1.0, 16. Jinja2==2.11.1. The text is color-coded: package names are in orange, version numbers are in green, and the equality sign is in blue. The background is dark grey.

```
1  asn1crypto==1.3.0
2  atomicwrites==1.3.0
3  attrs==19.3.0
4  certifi==2020.4.5.1
5  cffi==1.14.0
6  chardet==3.0.4
7  click==7.1.1
8  codecov==2.0.15
9  coverage==5.0
10 cryptography==2.8
11 Flask==1.1.2
12 Flask-Cors==3.0.8
13 idna==2.9
14 importlib-metadata==1.5.0
15 itsdangerous==1.1.0
16 Jinja2==2.11.1
```

This file consists of the sequence to sequence model architecture that has been built using Keras framework.

3. spellingcorrector.py

```
spellcorrector.py x
1 import pkg_resources
2 from symspellpy import SymSpell, Verbosity
3
4 sym_spell = SymSpell(max_dictionary_edit_distance=2, prefix_length=7)
5 dictionary_path = pkg_resources.resource_filename("symspellpy", "frequency_diction
6 bigram_path = pkg_resources.resource_filename("symspellpy", "frequency_bigramdict:
7
8 # term_index is the column of the term and count_index is the
9 # column of the term frequency
10 def spell_corrector(input_term):
11     sym_spell.load_dictionary(dictionary_path, term_index=0, count_index=1)
12     sym_spell.load_bigram_dictionary(bigram_path, term_index=0, count_index=2)
```

This file consists of the prediction process.

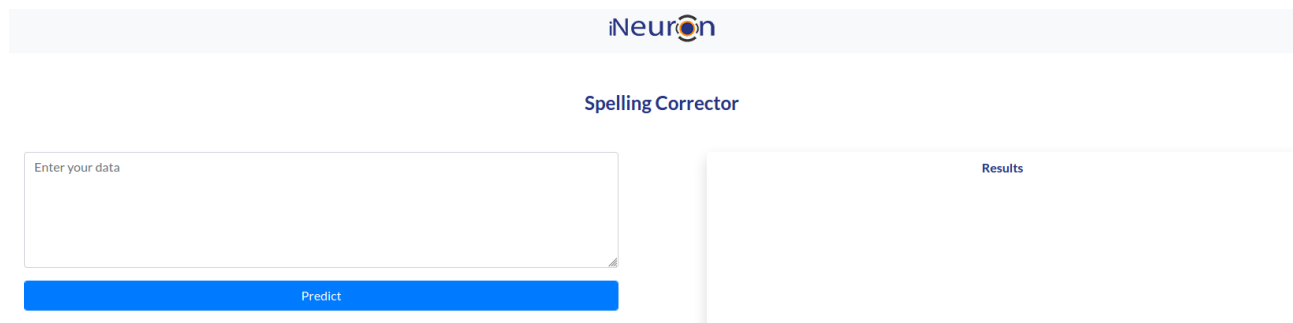
4. clientApp.py

```
clientApp.py x
1 from flask import Flask, request, jsonify, render_template
2 import os
3 from flask_cors import CORS, cross_origin
4
5 from spellcorrector import spell_corrector
6
7 os.putenv('LANG', 'en_US.UTF-8')
8 os.putenv('LC_ALL', 'en_US.UTF-8')
9
10 app = Flask(__name__)
11 CORS(app)
12
13 @app.route("/", methods=['GET'])
14 @cross_origin()
15 def home():
16     return render_template('index.html')
```

This file consists of flask and this is the entry point of application.

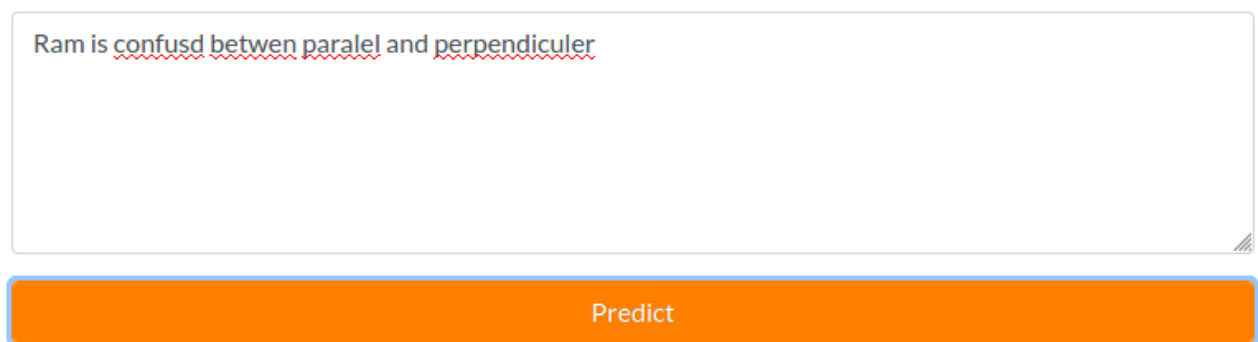
TESTING IN LOCAL/API

To do the test testing we need to run the `clientApp.py` and after that web server will start at <http://0.0.0.0:5000/>



The screenshot shows the iNeuron Spelling Corrector web application. At the top, there is a header with the iNeuron logo. Below the header, the title "Spelling Corrector" is displayed. The main interface consists of two columns. The left column has a text input area labeled "Enter your data" and a blue "Predict" button below it. The right column has a "Results" label at the top, followed by a large empty box for displaying the results.

Enter the wrong spelled sentence and click on predict.



The screenshot shows the iNeuron Spelling Corrector web application with a sentence entered in the input field: "Ram is confusd between paralel and perpendiculer". The words "confusd", "paralel", and "perpendiculer" are underlined with red wavy lines, indicating they are misspelled. Below the input field is an orange "Predict" button.

After clicking Predict

Spelling Corrector

Ram is ~~confusd~~ ~~between~~ ~~paralel~~ and ~~perpendicular~~

Predict

Results

"ram is confused between parallel and perpendicular"

Results

"ram is confused between parallel and perpendicular"

Finally the answers are shown in the results box.

CONCLUSION

Here we have successfully built a spelling checker and corrector which can be used to correct spelling if the user gives misspelled words.

COMPARISION

We can build better accurate models using pretrained models like BERT, GPT2 etc. Also here we have given very less training data but the user can increase the sizing of training data which will help to build a better model with better prediction.