ReadMe

Progress since the last session

1. We added sorting to dates, so it doesn't matter in which order the user writes them.
2. Added the possibility for empty input.
3. Changed the classification to compare the input with vectors for possible sentences for the category.
4. Added generation of check data inputs, by using a variable in the example sentences that gets substituted with each possible value.
5. Added the possibility for the user to ask for the repetition of a question.
6. Added a stop functionality, where the chatbot recognizes if the user requests to stop the conversation.
7. Changed the functionality of the bullet points for the part after Personal Information.
8. Added a history, to be able to handle requests to see the data of the previous question or stage.
9. Implemented the Skills Section with SpaCy.
10. Coloring for the Debug and Normal Chatbot output.
11. Changed the date parsing from SpaCy to regex, since SpaCy is really unreliable in that regard.
12. Added a real start and end of the conversation. With printing the full CV at the end as well.
13. We fixed some bugs we had with saving the address, the user gives the chatbot.
14. Fixed some issues with the recognition of categories in the bullet-point function.

Functionality the chatbot can handle

Generally each input can belong to one of four categories.
An answer to a question, a request for repetition, a request to stop or a request to display some already saved data.

Answer

Normal answer (belonging to Personal Information)

If no request is detected the input is handled as an answer. Then depending on the question the chatbot tries to parse
the data using SpaCy, regex, or/and datetime. If data is missing, it asks the user to supply it. Here the user once again,
can say something belonging to each of the four categories. If the answer isn't satisfactory the chatbot will continue to
ask, until it can fill the data slot. Because of that there will be a good CV without missing data as the final version.

Bullet Point answer (belonging to for example education)

Here it is basically the same, only with the added functionality to be able to continue to add bullet-points or stop.
The user can stop by inputting nothing, so enter.

What we handle with SpaCy/regex/datetime

Regex/datetime

We handle everything with regex/datetime, what SpaCy can't handle out of the box or only unreliably.
The address and the e-mail address is parsed with regex, since it is really specific.
For all dates we use datetime from python. Therefore, we have defined specific formats which we decided to work with.
For that please take a look at the utils Class and the "date_formats" stated there. The advantage over SpaCy is
that dates are better detected and stick with a consistent format which in the end does not lead to several date formats printed on the CV. Furthermore we have the advantage of sorting dates and therefore also display them in
the correct order no matter in which order the dates were given in the input.

Spacy

Here we mostly try to get the important entities, like Person or Organisation.
Currently, we are using SpaCy in all stages:

- Personal Data: detecting Names
- Education, Experience and Social Engagement: detecting Organizations
- Skills: detecting the different skills by using several optional Entities

Repetition

Here the chatbot will confirm, that it can repeat the question and then ask again. Here the input can once again belong
to each of the four categories.

Goodbye

If a request to stop is detected. The goodbye sequence is started, that also starts, when all questions have been answered.
Here the chatbot says goodbye and prints the whole CV.

Check Data

If this category is detected, the chatbot then continues to parse which data has been requested.

Possibilities are:
all data, data of the previous question or stage, data of a specific question or stage

Here the input is checked for possible variants of the question/stage names.
And then the corresponding data is printed.
If the requested data isn't found, the user has the option to try again or just leave it.

How to run

Packages we use (from all of them the most current version):

- datetime
- numpy
- sys
- spacy (installation guide https://spacy.io/usage)

- re

For SpaCy to make it easier to use, the current necessary data is downloaded on each run.
The best way to run it, is to use the console and typing "python main.py"
To use the debug features, the flags in utils.py have to be changed.

debug_info

If set to true, a lot of interesting information is displayed like the cosine similarities etc.

debug_text

To use it, the debug_key has to be set to either "Standard" or "Mixed". Which results in the inputs for the chatbot
coming from an already written dialog. The outputs of that are at the bottom of the ReadMe.
If it is set to false, input is requested in the command line.

Example Dialogs

For the example dialogues please view the attached files. We have created 4 different examples which all test and show
the stated features above. But of course also feel free to directly interact with our Bot by running the code yourself.

Experience with SpaCy

Last but not least we wanted to share our experience with SpaCy. We think that for several entities it does a great job
recognizing them, but also has a lot of weaknesses, e.g. when trying to filter Date Entities. Therefore, we tried to
implement our own approach using datetime from python. For our specific use case the results here turned out to be
better, and therefore we stuck with it.