Intention Parser v1.0:

1. Core functionality:

The main task of Parser is to get intention to obtain/receive/inquiry something and translate this intention into computer command (e.g. “What is the weather like today?” - Intent: Get Weather, “Tell me an interesting fact.” - Intent: Get Fact, etc.).

2. Implementation explained:

The coverage of the Parser is much wider than just treat key words as “weather” and “fact”. It deals with a lot of different intentions to get something. Moreover, negative examples also presence - an inquiry such as “The weather is bad”, “I was told something”, etc. would not be considered as intentions, the program will output “(Intent: -)” for similar cases.

Parser uses machine learning approach, in particular, NLP, to process every inquiry. Thus, making it fast running and scalable enough. Though, some parts of processing are supposed to be done by parsing it manually to maintain simplicity and matching pattern of a neural network. In current version city parsing is done manually.

City dictionary is loaded into RAM memory as a hash map (I decided that it might be useful in the future to get a code of a city for further processing, for instance, to really get a weather in the specified location, though, now program doesn’t make use of it). ***However, now this functionality is limited by treating one word cities or *-*-… like cities.***

3. Technical details:

In the world, approximately 2.7M cities exist, if we assume that average length of a city name is roughly 15 characters, so the memory needed to fit the dictionary is about 40 Mb. Though, key collisions must be considered more precisely (the speed should really matter for real application). In the case of poor performance a set of hash maps could be implemented: create mapping from any word to, lets say 10 maps containing cities - map[word] yields cityDict(n) -> cityDict(n).find(word).

Every inquiry gets slightly normalised - it transforms to lower case character and extra spaces are deleted.

Tuned GPT-3 model is used as a core for Parser. Though, another libraries such as spacy exist.

A python script called inside c++ code to make API call to GPT-3. Venv created and openai is downloaded in it.

In case of real application, I am sure that MBition has a department developing a NLP model. Thus, it would be relatively simple AI returning 2 words or one symbol answers that should be easy to train and get the desired accuracy.

Any other way of solving this issue seemed useless - we could simply get through string and find trigger words or use some sort of regular expression matching. But this approach is far from being scalable at all - to scale it up we would have to manually extend our dictionary of words or expression and we still might not cover every use case. Probably, there could be not so many possible phrases, though it might be above one thousand. It is neither nice to code nor to use, as the performance would get more and more poor with every case we add to our code leading to O(K * N) time complexity instead of O(N).

4. User Guide:

The program expect initially the full path to the city dictionary (which is now located at the root of the project folder). You will have 9 attempts to open file, otherwise program will stop (it is very interactive and will give you prompts).

Then a loop for inquires begins, Parser expects any string with only restriction mentioned above regarding the city (and the city must be on the list actually).

You will get response until you write “c” in command line to stop the execution.

Watch Video Guide attached!

Have a good day!