Task Sheet PR03
Stemming, Inverted List and Evaluation

Preliminary Notes:

All notes and requirements of PR02 still apply!

Task 1 – Stemming

1. Implement a stemming functionionality that uses the Porter algorithm by implementing all empty functions in the file `porter.py` in the template.
2. A description of all rules of the algorithm can be found in the file `porter.txt` (available in the `raw_data` folder of the template).
3. Although you may use other sources to understand the algorithm, you should still study the document carefully and make sure that your submission follows all of the rules listed! We will use that document to evaluate your solution.
4. Naturally, the use of methods from NLP libraries such as `nltk` is strictly prohibited! However, using `re` is permitted.
5. If the user chooses stemming for the search for a document, it should be done via *mapping to the stem form*.

Hints:

- Vocals are not just A, E, I, O, U but also Y, if a consonant comes before (e.g. like toy).
- If multiple rules apply with same ending: Only the rule with the longest matching ending is used (e.g. argument stays argument in step 4).
- Rule conditions on the *measure* are always based on the word *after* the ending was removed!

Task 2 – Inverted List for Boolean Retrieval

1. Implement an inverted list for the linear search! Enable the search using this inverted list by filling `inverted_list_search()` in `ir_system.py` with the proper code!
2. If the inverted list is not initialized, the function should first build it and then use it. In subsequent searches, the existing inverted list should then be used. However, it is not required that the inverted list persists among multiple program runs.
3. Enable the user to search using the conjunction, disjunction and negation of terms! Implement these operators for both the linear search and the inverted list!
4. The user should be able to use these operators directly in their query string. The syntax should be as follows (without spaces):

operation	syntax	example
conjunction	t_1&t_2	`fox&wolf`
disjunction	t_1|t_2	`fox|wolf`
negation	-t_1	`-fox`
grouping	(t_1)	`(fox)`

Parsing libraries may be used for processing the query.

Hints:

- `InvertedListBooleanModel` is similar to `LinearBooleanModel` in its matching and representation methods. Try to encapsulate as much of the inverted list processing logic in the class as possible.
- You may need multiple inverted lists for normal or stemmed words.
- Keep in mind that the Boolean operations correspond to set operations. Using this might make the implementation easier.

Task 3 – Evaluation

1. Implement the calculation of precision and recall for queries that contain the terms listed in the file *raw_data/ground_truth.txt* (to be found in the template)! To do this, fill the empty functions in `ir_system.py`!
2. For queries that consist of more than one search term from *ground_truth.txt*, the calculation should also work! If the calculation of precision and/or recall is not possible, the program must not crash! You may return a non-value, such as -1.
3. Add code to the `main_menu()` that measures the taken time for the query processing in ms. Print this value on the CLI via `print()` right under the precision/recall values!

Hints:

- Be careful about the document IDs in *ground_truth.txt* - they start with 1!
- If you commented out the respective function calls in PR02, do not forget to uncomment them again.

Make sure that your solution considers all requirements listed in this file and upload it on Moodle until the specified deadline!