

**Project Internship – Intelligent Agents**  
**CS4514-P – WS2023/24**

## **Exercise 2**

Magnus Bender, Ralf Möller

Published: 17.11., 12:00

Presentation: 15.12., 12:15 - 13:45

### **Preface**

The purpose of this internship is the implementation of an Information Retrieval (IR) agent based on a collection of Wikipedia articles. The agent answers queries by returning Wikipedia articles from its corpus. Besides answering queries, the agent is also interested in maintaining its corpus, e.g., by extending it with interesting new articles.

In the last exercise, we have implemented two different types of agents (i.e., auctioneer and bidder) performing an auction of documents from Wikipedia articles. In this exercise, we enhance the agents' functionalities and provide the IR part, i.e., the query answering. The previous bidder agent has to be extended with the additional functionality of answering queries. Furthermore, a new questioner agent, asking queries, has to be created.

We apply techniques introduced in the lecture *Web Mining Agents* to select the relevant documents. In the end, answering queries and running the auction should be possible altogether.

**Hint 1** *We have updated the Project Package in the Moodle course for exercise 2.*

### **Query Answering**

In the last exercise, we used the simple *tf.idf*-score, in this exercise we will use Latent Dirichlet Allocation (LDA). The LDA topic model is used to represent the topics of the IR agent's corpus. Given a query, again a Wikipedia article, the agent returns the most relevant Wikipedia articles from its corpus.

**Hint 2** *The auction part may still use the *tf.idf*-score, but a group may also choose to use LDA.*

## Task 1: Latent Dirichlet Allocation

In the first task, we (i) learn how to build a topic model from Wikipedia articles, (ii) understand the influence of hyperparameters on an LDA topic model, (iii) visualize the LDA topic model.

The following four files are available in the *Project Package* in the directory **src/exercise2/data**. As before, each file contains the titles of the Wikipedia articles to use.

**corpus.txt** The initial corpus of each IR agent.

**sell.txt** The list of articles to sell by the auctioneer.

**queries.i.txt, queries.ii.txt** The list of questions to be asked by the questioner to the IR agent. (There will be two IR agents starting with the same corpus, but getting different questions each.)

Perform the following steps to build the corpus which is used for all tasks of this exercise:

- (i) Download for each title in the four **\*.txt** files the corresponding article. Make sure to store the articles separately and grouped by their originating file.
- (ii) Preprocess the text of each article as in the last exercise, but use a lemmatizer instead of a stemmer (also provided by NLTK).
- (iii) Prepare the documents for the LDA topic model by computing for each article bigrams and adding them to the corresponding document.

Only add those that occur more than 20 times in the documents and replace white space between two words with an underscore.

**Question 1** *Describe a scenario where bigrams or even trigrams can increase the performance of LDA.*

**Question 2** *Why should one use a lemmatizer instead of a stemmer for preprocessing for the computation of an LDA model?*

Use the Python package **gensim**<sup>1</sup> to implement LDA, as the package contains all necessary functions for this exercise. It is already installed in our Docker-Image. The following three parameters affect the LDA model and thus the performance of the IR-agent:

$\alpha$  The Dirichlet prior on the per-document topic distributions

$\beta$  The parameter of the Dirichlet prior on the per-topic word distribution

$K$  The number of topics

---

<sup>1</sup><https://radimrehurek.com/gensim/models/ldamodel.html>

**Hint 3** Remember that the corpus changes after buying a document in the auction, which means that the LDA topic model might have to be re-trained.

The LDA model can be evaluated to find the best values for  $\alpha$ ,  $\beta$ , and  $K$ . Generally, the evaluation can be done in an intrinsic and an extrinsic way.

- (i) Create topic models for all documents using `gensim` with different values for  $\alpha$ ,  $\beta$ , and  $K$ . It might be helpful to train models on subsets of all documents, e.g., only on the initial corpus or on the query documents and the initial corpus together.
- (ii) Compute the *perplexity* and *coherence score*<sup>2</sup> of the topic models to identify the best values for the individual collection of documents.
- (iii) Use the Python package `pyLDavis`<sup>3</sup> to visualize the resulting topic model with the best values, in form of a word cloud for each topic<sup>4</sup>.

**Hint 4** Think about which subsets of all documents will later be included in the IR agent's corpus. The hyperparameters should be chosen accordingly.

**Question 3** Explain the differences between an intrinsic and extrinsic evaluation and think about the use cases for both evaluation methods.

**Question 4** The hyperparameters  $\alpha$  and  $\beta$  are a trade off between two conflicting goals to find groups of tightly co-occurring words. Describe both goals and explain why those goals are conflicting.

**Question 5** How many topics would you expect from the documents in the corpora and how many does your best model contain?

## Task 2: Performance of an IR-Agent

In the last task, we trained an LDA topic model to represent the corpus of the IR agent. However, we did not implement the IR part yet. Generally, Wikipedia articles are used as queries and the aim is to return similar documents from the agent's corpus.

Since each document is represented by a distribution over  $K$  topics, the query (which is a document sent by the user to the agent) can be represented in the same way and the documents in the corpus most similar to the query can be identified. Use the *Hellinger distance* to calculate the distance between the topic distribution of the query and the topic distribution of documents in the corpus. Use the articles from `queries.i.txt` or `queries.ii.txt` as queries. Finally, it is possible to perform an extrinsic evaluation by analyzing the information retrieval performance of the agent.

---

<sup>2</sup>Both values may be calculated using `gensim`.

<sup>3</sup><https://github.com/bmabey/pyLDavis>

<sup>4</sup>The Docker-Image does not provide a graphics driver, thus, use the `pyLDavis.save_html()` function to save output as an HTML file after loading the model using `pyLDavis.gensim_models`.

**Question 6** *There are queries that are not covered by the corpus, i.e., the topic of Moodle does not match any document in the corpus. What should be done in such cases? (considered more closely in Task 3)*

**Hint 5** *To get the document topic distribution of the query, a document not contained in the corpus, you might use FIGS [KBBM20]. In such cases, FIGS is used automatically by the function `get_document_topics`.*

- (i) Compute *precision*, *recall*, and the *F1-score*<sup>5</sup>.
- (ii) Compare the results of tf.idf and LDA for IR. (This implies that your code is able to answer queries using tf.idf and LDA.)

**Question 7** *Are there any problems regarding the creation of the gold standard?*

Since we use a distance to identify the best matching documents to a query, we receive for each query an answer as a set of documents. However, for some queries, the agent has not really a *correct* answer resulting in a large distance of the best matching documents. The documents the agent returns are not those the human would expect, as they have a high distance to the query.

In the next tasks, we focus on a human-aware behaviour of the agent. The agent should recognize its inability of answering the query in a way desired by the human and the agent should be able to think about the model the human might have in mind (thus the answer the user is expecting) when stating a query. Then, the agent can compare the model of the human with its own model.

### Task 3: Explicable and Explainable Behaviour

In this task, we are interested in a human-aware behaviour of the agent. Figure 1 represents the following models:

- $M^H$  is the mental model of the human and in this case a set of documents the human would like to retrieve.
- $M_r^H$  is the mental model the agent approximates that the human has and in this case a set of documents the human would like to retrieve.
- $M_h^R$  is the mental model the human has about the agent and in this case a set of documents the human expects that the agent returns.
- $\widetilde{M}_h^R$  is the mental model the agent approximates that the human has about the agent and in this case a set of documents the human expects the agent would return.
- $M^R$  is the mental model of the agent and in this case a set of documents the human would like to retrieve.

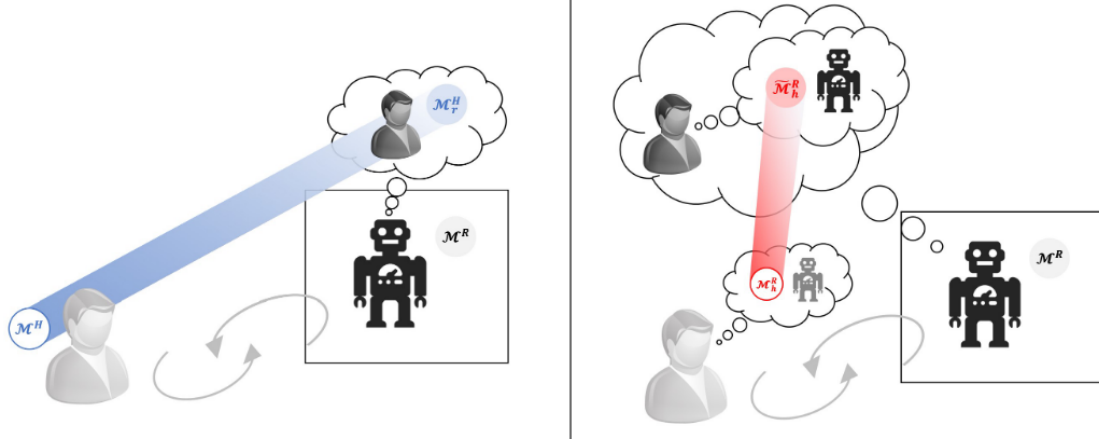


Figure 1: Human-Aware AI Models [Kam19].

The goal of this task is to go towards a human-aware agent as depicted in Figure 1. The agent should try to anticipate the humans information need and expectations.

- (i) Implement a function that returns the discrepancy of the model of the user  $M_r^H$  with the model of the agent  $M^R$ . Specifically, the function should return {Compatible, Moderate, Incompatible}, along with either matching results or the information that the agent cannot answer the query with its corpus.

The function should have the following properties:

**Compatible** The highest ranked documents have a high score and thus the agent can assume that the results are those the human would expect.

⇒ The agent returns results in the same way as usual.

**Moderate** The highest ranked documents have a moderate score and the agent is unsure whether the results are those the human would expect.

⇒ The agent considers the tf.idf-scores of the documents as a source of more information and uses a combination of both scores for the decision.

**Incompatible** The highest ranked documents have a low score and the agent can assume that it is not able to return documents the user would expect.

⇒ The agent gives some examples from its corpus (e.g., titles of articles) and explains that the user should send the query to another agent.

The parameters for differentiating between this three cases should be chosen appropriately.

---

<sup>5</sup>Create the gold standard by yourself, e.g., by defining good answers for each of the queries.

**Hint 6** *In case of moderate, sort the best documents for the LDA model and the best documents for tf.idf in descending order. Add the ranks of both for each document. Return the documents with the lowest overall rank.*

**Question 8** *What could your agent alternatively do in case of moderate?*

## Task 4: Extended Communication Agents

In this task, we combine the solution of exercise 1 with the solutions we have just developed.

**Hint 7** *If a new document extends the corpus, the LDA topic model of the agent needs to be updated [KBBM20]. However, this might be done batchwise to speed the agent up.*

The solution of this exercise shall consist of five agents of three types:

**1x Auctioneer** The same agent as in exercise 1, sells the documents to the IR agents.

**2x Questioner** Asks queries to an IR agent. The first questioner uses the queries from **queries\_i.txt** and asks them to the first IR agent, while the second questioner asks the **queries\_ii.txt** to the second IR agent.

**2x IR Agent** Combines the bidding functionality and the answering of queries. Initially, both agents start with the same corpus and no queries. The auction will run simultaneously along with the query answering. Thus, the different queries from the questioners form the different topics of each IR agent during the auction.

**Hint 8** *The queries sent by the questioners should influence the values of the bids given in the auction.*

**Hint 9** *A SPADE agent is a stateful system. Thus, think about synchronization of the agents, e.g., what happens if the agent has just submitted a bid and then receives a new query before the winner of the document has been announced?*

*The auction and query answering does not need to run in parallel. Also, it might be an idea to add a query type in each of the sent messages to ensure that the system is able to handle unsynchronized queries.*

The solution of this exercise has to contain a Python file starting all five agents. The agents shall generate output to show what is going on. Additionally, please add one of your HTML files generated by `pyLDavis` along with the selected hyperparameters.

## Outlook

We have created an IR agent, answering queries based on its corpus and also bidding in an auction to extend its corpus.

In the next exercise, the bidding strategy will be improved in the way that one agent tries to anticipate the bidding of the other agent and improves its strategy in this way. Further, the questioners might send the queries to more than one agent.

## References

- [Kam19] KAMBHAMPATI, Subbarao: Challenges of Human-Aware AI Systems. In: *arXiv preprint arXiv:1910.07089* (2019)
- [KBBM20] KUHR, Felix ; BENDER, Magnus ; BRAUN, Tanya ; MÖLLER, Ralf: Maintaining Topic Models for Growing Corpora. In: *Proceedings of the 14th IEEE International Conference on Semantic Computing (ICSC-20)* (2020), 451–458. [https://www.ifis.uni-luebeck.de/uploads/tx\\_wapublications/Maintaining\\_Topic\\_Models\\_for\\_Growing\\_Corpora\\_public.pdf](https://www.ifis.uni-luebeck.de/uploads/tx_wapublications/Maintaining_Topic_Models_for_Growing_Corpora_public.pdf)