NLP and the Web – WS 2024/2025

Lecture 14
Exam Preparation

Dr. Thomas Arnold
Hovhannes Tamoyan
Kexin Wang

Ubiquitous Knowledge Processing Lab
Technische Universität Darmstadt

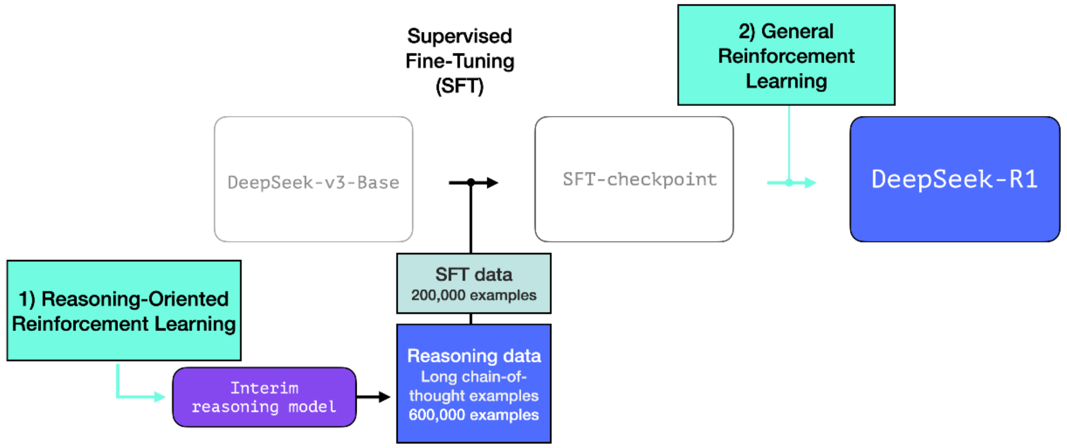
Syllabus (tentative)

Nr.	Lecture
01	Introduction / NLP basics
02	Foundations of Text Classification
03	IR – Introduction, Evaluation
04	IR – Word Representation
05	IR – Transformer/BERT
06	IR – Dense Retrieval
07	IR – Neural Re-Ranking
08	LLM – Language Modeling Foundations, Tokenization
09	LLM – Neural LLM
10	LLM – Adaptation
11	LLM – Prompting, Alignment, Instruction Tuning
12	LLM – Long Contexts, RAG
13	LLM – Scaling, Computation Cost
14	**Review & Preparation for the Exam**

DeepSeek

Very nice blog about the new DeepSeek model

https://newsletter.languagemodels.co/p/the-illustrated-deepseek-r1


Supervised Fine-Tuning (SFT)
2) General Reinforcement Learning
DeepSeek-v3-Base
SFT-checkpoint
DeepSeek-R1
SFT data 200,000 examples
1) Reasoning-Oriented Reinforcement Learning
Interim reasoning model
Reasoning data Long chain-of-thought examples 600,000 examples

General info about the exam

- Modus: Written close-book exam in Darmstadt (in-person)
- Date: 25.02.2025
- Time slot: 15:00 – 17:00
- Where: Will be announced on Moodle (~1 week before the exam)

General info about the exam

- No books, notes, or other auxiliary material may be used
- For math problems you can use **non-programmable** calculator.
- Problems are **stated in English**
- The questions may be **answered in either German or English**.
- You will have ~90-100 minutes to complete the exam

Questions

What kind of tasks can we expect in the exam? (multiple-choice / open questions ...)

Answer:

- ~10% "Know stuff", examples:
  - Definitions of basic terms (What is a morpheme?)
  - Remember lecture topics (Name 2 approaches for parameter-efficient fine-tuning)

- ~30% "Understand stuff", examples:
  - Compare metrics / methods (Why is F1 better than Accuracy in IR?)
  - Explain a method (What is the key difference of decoder self-attention compared to encoder self-attention?)

Questions

What kind of tasks can we expect in the exam? (multiple-choice / open questions ...)

Answer (continued):

- ~30% "Do stuff", examples:
  - Tokenization, TF-IDF, inverted index, Viterbi, Precision/Recall, ranked evaluation metrics, Byte-Pair Encoding…

- ~30% "Transfer knowledge", examples:
  - Here is a scenario X. Would you use an encoder or decoder transformer model? Why?
  - Why is tokenization especially hard in Twitter data?

- Max. one multiple-choice questions

Hint for practicing

LLMs are very helpful for practice!

Example:

- Import a slide PDF into ChatGPT, ChatPDF, or a similar tool
- Simple prompt: “Create 10 example exam questions on the topics of this slide package. The questions should range from simple knowledge or understanding questions to more challenging transfer questions.”
- You can also let the LLM answer the questions and **critically** compare with your answer (it might give wrong answers!)

Know Stuff / Understand Stuff / Transfer Knowledge

Q: What is a morpheme? Give a word that consists of multiple morphemes as an example and mark its morphemes.

Q: What is the difference between intrinsic evaluation and extrinsic evaluation? Give an example for each.

Know Stuff / Understand Stuff / Transfer Knowledge

Q: What is a morpheme? Give a word that consists of multiple morphemes as an example and mark its morphemes.

A: A morpheme is the smallest meaningful unit of language. Example: "unhappiness" → un- (prefix, negation) + happy (root, adjective) + -ness (suffix, noun-forming).

Q: What is the difference between intrinsic evaluation and extrinsic evaluation? Give an example for each.

Know Stuff / Understand Stuff / Transfer Knowledge

Q: What is a morpheme? Give a word that consists of multiple morphemes as an example and mark its morphemes.

A: A morpheme is the smallest meaningful unit of language. Example: "unhappiness" → un- (prefix, negation) + happy (root, adjective) + -ness (suffix, noun-forming).

Q: What is the difference between intrinsic evaluation and extrinsic evaluation? Give an example for each.

A:

Intrinsic evaluation measures performance on a specific task using direct metrics (e.g., perplexity for language models). Example: Evaluating a word embedding model with cosine similarity on word similarity tasks.

Extrinsic evaluation assesses usefulness in an end application. Example: Measuring how different embeddings affect machine translation accuracy.

Know Stuff / Understand Stuff / Transfer Knowledge

Q: N-Gram language models can run into problems of zero probabilities when dealing with OOV tokens. Name and briefly describe two methods that can deal with these problems.

Know Stuff / Understand Stuff / Transfer Knowledge

Q: N-Gram language models can run into problems of zero probabilities when dealing with OOV tokens. Name and briefly describe two methods that can deal with these problems.

A:

Smoothing (e.g., Add-one smoothing, Laplace-smoothing): Assigns small probabilities to unseen n-grams to avoid zero probabilities.

Backoff & Interpolation: Uses lower-order n-grams when higher-order ones are missing, combining probabilities from different n-gram levels.

Know Stuff / Understand Stuff / Transfer Knowledge

Q: What is the goal of Instruction Tuning in neural LLMs? What data do you need?

Q: What are the main components of Hidden Markov Models (HMM)?

Know Stuff / Understand Stuff / Transfer Knowledge

Q: What is the goal of Instruction Tuning in neural LLMs? What data do you need?

A: The goal of Instruction Tuning is to improve a language model's ability to follow diverse and complex instructions by fine-tuning it on task-specific prompts and responses.

Data needed: A dataset of instruction-response pairs covering various tasks, often human-annotated or synthesized.

Q: What are the main components of Hidden Markov Models (HMM)?

Know Stuff / Understand Stuff / Transfer Knowledge

Q: What is the goal of Instruction Tuning in neural LLMs? What data do you need?

A: The goal of Instruction Tuning is to improve a language model's ability to follow diverse and complex instructions by fine-tuning it on task-specific prompts and responses.

Data needed: A dataset of instruction-response pairs covering various tasks, often human-annotated or synthesized.

Q: What are the main components of Hidden Markov Models (HMM)?

A:

States: Hidden variables representing underlying conditions.

Observations: Visible outputs generated from states.

Transition probabilities: Probabilities of moving between states.

Emission probabilities: Probabilities of observations given a state.

Initial state distribution: Probabilities of starting in each state.

Know Stuff / Understand Stuff / Transfer Knowledge

Q: What is the difference between an information need and a query? Give an example.

Know Stuff / Understand Stuff / Transfer Knowledge

Q: What is the difference between an information need and a query? Give an example.

A: An information need is a user's underlying goal, while a query is the specific formulation of that need in a search system.

Example:

Information need: "I want to find the best hiking trails near me.“

Query: "best hiking trails in Darmstadt."

Know Stuff / Understand Stuff / Transfer Knowledge

Q: What is the main advantage of word embeddings over one-hot encoding in Information Retrieval?

Know Stuff / Understand Stuff / Transfer Knowledge

Q: What is the main advantage of word embeddings over one-hot encoding in Information Retrieval?

A: Word embeddings capture semantic similarity, allowing words with similar meanings to have close vector representations, whereas one-hot encoding treats each word as independent and equal-distant, ignoring relationships between words.

Also:

- Reduced dimensionality
- Better usage of memory (dense embeddings vs. sparse one-hot vectors)
- …

Know Stuff / Understand Stuff / Transfer Knowledge

Q: Compare and contrast Recurrent Neural Networks (RNNs) and Transformers for processing text sequences. In what situations would an RNN still be preferable?

Know Stuff / Understand Stuff / Transfer Knowledge

Q: Compare and contrast Recurrent Neural Networks (RNNs) and Transformers for processing text sequences. In what situations would an RNN still be preferable?

A: RNNs process sequences step by step, maintaining hidden states and shared weights, but suffer from slow training (and vanishing gradients).

Transformers use self-attention to process all tokens in parallel, making them more efficient for long-range dependencies.

RNNs may still be preferable in resource-constrained settings due to their lower memory requirements, or for online sequence processing where inputs arrive sequentially.

Know Stuff / Understand Stuff / Transfer Knowledge

Q: Explain the role of convolutional neural networks (CNNs) in modeling word n-grams.
How does a 1D CNN process textual input?

Know Stuff / Understand Stuff / Transfer Knowledge

Q: Explain the role of convolutional neural networks (CNNs) in modeling word n-grams.

How does a 1D CNN process textual input?

A: CNNs capture local patterns, such as word n-grams, by applying filters over sequences of word embeddings. A 1D CNN processes text by sliding convolutional filters over word vectors, detecting features like key phrases. The results pass through activation functions and pooling layers to reduce dimensionality and highlight the most important n-gram features.

Know Stuff / Understand Stuff / Transfer Knowledge

Q: Given a document retrieval system, how would you incorporate BERT to improve ranking? What modifications would be necessary to fine-tune BERT for this purpose?

Know Stuff / Understand Stuff / Transfer Knowledge

Q: Given a document retrieval system, how would you incorporate BERT to improve ranking? What modifications would be necessary to fine-tune BERT for this purpose?

A:

Use BERT for re-ranking: Apply it to rerank top candidate documents retrieved by a traditional system (e.g., BM25).

Fine-tuning: Train BERT on query-document pairs using relevance labels. Potentially distill BERT into a smaller model for scalability.

Know Stuff / Understand Stuff / Transfer Knowledge

Q: What are the advantages and disadvantages of using a mono-stage vs. a duo-stage ranking approach in neural IR?

Know Stuff / Understand Stuff / Transfer Knowledge

Q: What are the advantages and disadvantages of using a mono-stage vs. a duo-stage ranking approach in neural IR?

A: Mono-stage ranking (e.g., dense retrieval) is faster and more scalable, as it retrieves and ranks documents in one step but may lack deep relevance understanding.

Duo-stage ranking first retrieves candidates (e.g., BM25, dense retrieval) and then re-ranks them with a more complex model (e.g., BERT). It is more accurate but computationally expensive.

Trade-off: Mono-stage is preferable for efficiency; duo-stage is better for precision-critical tasks.

Know Stuff / Understand Stuff / Transfer Knowledge

Q: Imagine you are designing a re-ranking system for a legal document retrieval task. Discuss two specific challenges that you expect compared to general web search, and two approaches how you might adapt neural re-ranking models to this domain.

Know Stuff / Understand Stuff / Transfer Knowledge

Q: Imagine you are designing a re-ranking system for a legal document retrieval task. Discuss two specific challenges that you expect compared to general web search, and two approaches how you might adapt neural re-ranking models to this domain.

A: Challenges:

Complex, domain-specific language: Requires specialized embeddings.

Long documents: Handling context efficiently is crucial.

(High precision need: Minor wording differences can change legal meanings.)

Adaptations:

Fine-tune on annotated legal datasets or use domain-adapted BERT models (e.g., LegalBERT).

Implement long-context processing (e.g., sparse attention patterns, sliding window approaches).

Know Stuff / Understand Stuff / Transfer Knowledge

Q: Discuss how HNSW creates a hierarchical structure for indexing data. What are the advantages of using multiple layers in the search process?

Know Stuff / Understand Stuff / Transfer Knowledge

Q: Discuss how HNSW creates a hierarchical structure for indexing data. What are the advantages of using multiple layers in the search process?

A:

HNSW (Hierarchical Navigable Small World) indexing builds a multi-layer graph where upper layers contain fewer, more connected nodes, enabling fast approximate nearest neighbor (ANN) search.

Advantages:

Logarithmic search complexity: Upper layers guide efficient traversal.

Scalability: Handles large datasets with low query latency.

Balanced exploration-exploitation: Ensures both fast access to likely matches and refinement at deeper layers.

Do Stuff

Q: Two information retrieval systems (System A and System B) are compared on a small test dataset with 2000 documents. For a query q, a total of 100 were annotated as being relevant to the query. Using this query, System A retrieves 200 documents, but 150 of them are marked as irrelevant. System B returns 80 relevant and 80 irrelevant documents.

Calculate Precision and Recall for each System.

Do Stuff

Q: Two information retrieval systems (System A and System B) are compared on a small test dataset with 2000 documents. For a query q, a total of 100 were annotated as being relevant to the query. Using this query, System A retrieves 200 documents, but 150 of them are marked as irrelevant. System B returns 80 relevant and 80 irrelevant documents.

Calculate Precision and Recall for each System.

A:

Precision (System A) = 50 / 200 = 0.25

Recall (System A) = 50 / 100 = 0.5

Precision (System B) = 80 / 160 = 0.5

Recall (System B) = 80 / 100 = 0.8

Do Stuff

Q: Given a corpus with the following three documents:

D1 = "Artificial intelligence is transforming computer science.“

D2 = "Machine learning is a subfield of artificial intelligence.“

D3 = "Computer science includes many subfields, including machine learning.“

(a) Compute the term frequency (TF) for the word "computer" in each of the three documents.

(b) Compute the inverse document frequency (IDF) for the word "computer".

(c) Determine the TF-IDF value for "computer" in the first document, using standard logarithmic weighting for TF and IDF.

A:

Do Stuff

A:

(a) TF(“computer”, Doc1) = 1, TF(“computer”, Doc2) = 0, TF(“computer”, Doc3) = 1

(b) “computer” appears in 2 out of 3 documents -> DF(“computer”) = 2.

IDF = log (N / DF) = log (3/2) ~ 0.176

(c) TF-IDF(“computer”, Doc1) =

log (1 + TF(“computer”, Doc1)) * IDF(“computer”) =

log (2) * log(3/2) ~ 0.053

Do Stuff

Q: A search system returns the following ranked list of documents for a user query:

System output: D1, D2, D3, D4, D5

Known relevant documents: D2, D4, D6

(a) Compute the Mean Reciprocal Rank (MRR) for this ranking.

(b) Compute the Average Precision (AP) for this ranking, assuming there are three relevant documents.

Do Stuff

Q: A search system returns the following ranked list of documents for a user query:

System output: D1, D2, D3, D4, D5

Known relevant documents: D2, D4, D6

(a) Compute the Mean Reciprocal Rank (MRR) for this ranking.

(b) Compute the Average Precision (AP) for this ranking, assuming there are three relevant documents.

A:

(a) The first relevant document in the list is D2 at rank 2. MRR = 1 / 2

(b) (1/2 + 2/4 + 0) / 3 = 1/3

Do Stuff

Q: System output: D1, D2, D3, D4, D5

(c) Compute nDCG@5, given the following relevance scores for the documents:

D1 : 1 D2 : $log_2(3)$ D3 : 6 D4 : 0 D5 : 0

Hint: Use $DCG@k = \sum_{i=1}^{k} \frac{relevance(i)}{log_2(i+1)}$ and normalize with $IDCG@5$

Do Stuff

Q: System output: D1, D2, D3, D4, D5

(c) Compute nDCG@5, given the following relevance scores for the documents:

D1 : 1 D2 : $log_2(3)$ D3 : 6 D4 : 0 D5 : 0

Hint: Use $DCG@k = \sum_{i=1}^{k} \frac{relevance(i)}{log_2(i+1)}$ and normalize with $IDCG@5$

A:

$$DCG@5 = \frac{1}{log_2(2)} + \frac{log_2(3)}{log_2(3)} + \frac{6}{log_2(4)} + \frac{0}{log_2(5)} + \frac{0}{log_2(6)} = \frac{1}{1} + 1 + \frac{6}{2} = 5$$

$$IDCG@5 = \frac{6}{log_2(2)} + \frac{log_2(3)}{log_2(3)} + \frac{1}{log_2(4)} + \frac{0}{log_2(5)} + \frac{0}{log_2(6)} = \frac{6}{1} + 1 + \frac{1}{2} = 7.5$$

$$nDCG@5 = \frac{5}{7.5} = \frac{2}{3}$$

Do Stuff

Q: Given a Hidden Markov Model (HMM) with two states S1 and S2, and transition probabilities:

P(S1|S1) = 0.5, P(S2|S1) = 0.5

P(S1|S2) = 0.4, P(S2|S2) = 0.6

The emission probabilities for observations O=(o1,o2) are:

P(o1|S1) = 0.8, P(o1|S2) = 0.1

P(o2|S1) = 0.2, P(o2|S2) = 0.9

Assume the initial probabilities are P(S1)=0.5 and P(S2)=0.5.

(a) Apply the **Viterbi algorithm** to compute the most likely sequence of states for the observations O=(o1,o2). Provide the complete Viterbi table (i.e., the computed probabilities at each step).
(b) What is the most probable state sequence?

A:

Do Stuff

A: Transition: P(S1|S1) = 0.5, P(S2|S1) = 0.5, P(S1|S2) = 0.4, P(S2|S2) = 0.6

Emission: P(o1|S1) = 0.8, P(o1|S2) = 0.1, P(o2|S1) = 0.2, P(o2|S2) = 0.9

Initial: P(S1) = 0.5 and P(S2) = 0.5

(a)

	o1	o2
S1		
S2		

Do Stuff

A: Transition: P(S1|S1) = 0.5, P(S2|S1) = 0.5, P(S1|S2) = 0.4, P(S2|S2) = 0.6

Emission: P(o1|S1) = 0.8, P(o1|S2) = 0.1, P(o2|S1) = 0.2, P(o2|S2) = 0.9

Initial: P(S1) = 0.5 and P(S2) = 0.5

(a)

	o1	o2
S1	0.5 * 0.8 = 0.4	
S2	0.5 * 0.1 = 0.05	

Do Stuff

A: Transition: P(S1|S1) = 0.5, P(S2|S1) = 0.5, P(S1|S2) = 0.4, P(S2|S2) = 0.6

Emission: P(o1|S1) = 0.8, P(o1|S2) = 0.1, P(o2|S1) = 0.2, P(o2|S2) = 0.9

Initial: P(S1) = 0.5 and P(S2) = 0.5

(a)

	o1	o2
S1	0.4	
S2	0.05	

Do Stuff

A: Transition: P(S1|S1) = 0.5, P(S2|S1) = 0.5, P(S1|S2) = 0.4, P(S2|S2) = 0.6

Emission: P(o1|S1) = 0.8, P(o1|S2) = 0.1, P(o2|S1) = 0.2, P(o2|S2) = 0.9

Initial: P(S1) = 0.5 and P(S2) = 0.5

(a)

	o1	o2
S1	0.4	
S2	0.05	

= max {
P (o2 | S1) * P (S1 | S1) * 0.4,
P (o2 | S1) * P (S1 | S2) * 0.05
}

= max {
P (o2 | S2) * P (S2 | S1) * 0.4,
P (o2 | S2) * P (S2 | S2) * 0.05
}

Do Stuff

A: Transition: P(S1|S1) = 0.5, P(S2|S1) = 0.5, P(S1|S2) = 0.4, P(S2|S2) = 0.6

Emission: P(o1|S1) = 0.8, P(o1|S2) = 0.1, P(o2|S1) = 0.2, P(o2|S2) = 0.9

Initial: P(S1) = 0.5 and P(S2) = 0.5

(a)

	o1	o2
S1	0.4	
S2	0.05	

= max {
0.2 * 0.5 * 0.4,
0.2 * 0.4 * 0.05
}

= max {
0.9 * 0.5 * 0.4,
0.9 * 0.6 * 0.05
}

Do Stuff

A: Transition: P(S1|S1) = 0.5, P(S2|S1) = 0.5, P(S1|S2) = 0.4, P(S2|S2) = 0.6

Emission: P(o1|S1) = 0.8, P(o1|S2) = 0.1, P(o2|S1) = 0.2, P(o2|S2) = 0.9

Initial: P(S1) = 0.5 and P(S2) = 0.5

(a)

	o1	o2
S1	0.4	
S2	0.05	

= max {
0.2 * 0.5 * 0.4 = 0.04,
0.2 * 0.4 * 0.05 = 0.004
}

= max {
0.9 * 0.5 * 0.4 = 0.18,
0.9 * 0.6 * 0.05 = 0.027
}

Do Stuff

A: Transition: P(S1|S1) = 0.5, P(S2|S1) = 0.5, P(S1|S2) = 0.4, P(S2|S2) = 0.6

Emission: P(o1|S1) = 0.8, P(o1|S2) = 0.1, P(o2|S1) = 0.2, P(o2|S2) = 0.9

Initial: P(S1) = 0.5 and P(S2) = 0.5

(a)

	o1	o2
S1	0.4	0.04 S1
S2	0.05	**0.18 S1**

= max {
0.2 * 0.5 * 0.4 = 0.04,
0.2 * 0.4 * 0.05 = 0.004
}

= max {
0.9 * 0.5 * 0.4 = 0.18,
0.9 * 0.6 * 0.05 = 0.027
}

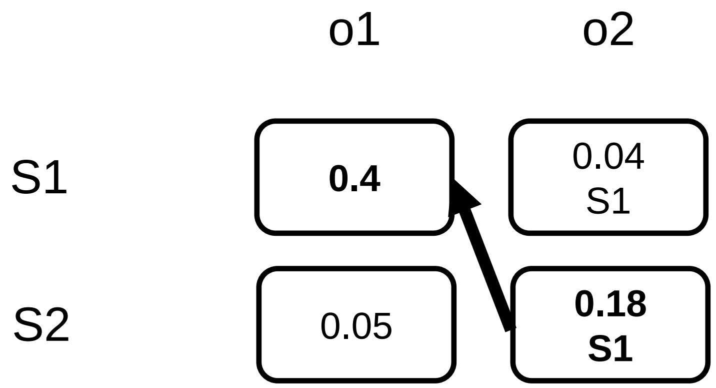
Do Stuff

A: Transition: P(S1|S1) = 0.5, P(S2|S1) = 0.5, P(S1|S2) = 0.4, P(S2|S2) = 0.6

Emission: P(o1|S1) = 0.8, P(o1|S2) = 0.1, P(o2|S1) = 0.2, P(o2|S2) = 0.9

Initial: P(S1) = 0.5 and P(S2) = 0.5

(a)



(b) The most likely state sequence for the given observations is S1, S2

Next Stop

Exam