Natural language processing: Question answering

The problem: question answering

- Question answering is the task of providing an answer to a question posed in natural language.
- Type of answer:
  - Multiple choice: the answer is selected from a predefined set of options (e.g., "What is the capital of France? A) Paris B) London C) Rome D) Berlin")
  - Open-ended: the answer is generated in natural language (e.g., "What is the meaning of life?")
- Types of question answering:
  - Closed-domain: questions about a specific topic or domain (e.g., "What is the capital of France?")
  - Open-domain: questions about any topic (e.g., "What is the meaning of life?")

Question answering

- **Inputs**: A question in natural language, and optionally some context or background information (e.g., a passage of text, a knowledge graph, etc.)
- **Outputs**: The answer to the question, either as a selected option or generated text.

Question answering benchmarks

- There are a large number of benchmarks for question answering, which can be categorized based on the type of question and the type of answer:
  - Multiple choice: e.g., SQuAD
  - Open-ended: e.g., Natural Questions, TriviaQA, etc.
- There are also benchmarks that focus on specific domains (e.g., biomedical question answering) or specific types of questions (e.g., commonsense question answering).

Where does the knowledge come from?

- Sometimes the answer is explicitly provided in the input (e.g., you provide a passage of text and then ask a question about it).
    - This is essentially a reading comprehension task.
- Sometimes the answer is not explicitly provided in the input, and the model has to rely on its internal knowledge (e.g., "What is the capital of France?").
    - This is essentially a knowledge-based question answering task.
    - So the model must have this information ahead of time.
- Sometimes the model is allowed to access external resources (e.g., a search engine, a knowledge graph, etc.) to find the answer.

What is truth?

- Who is Luke Skywalker's father?
  - At the end of the first Star Wars movie, it is a Jedi Knight who was killed by Darth Vader.
  - At the end of the second Star Wars movie, it is revealed that Darth Vader is Luke Skywalker's father.
  - In the real world, there is no such person as Luke Skywalker, so the question is essentially meaningless.
  - So the answer to this question depends on the context and the time at which it is asked.

How it works: let's hack a question answering model

- Q: What is the capital of France?
- We create a template: "The capital of France is [MASK]."
- We perform a Google search for this. Record the top 100 webpages that are returned.
- Count the words that appear:
  - Paris: 90
  - hosting: 7 (hosting the Olympics)
  - awaiting: 3 (the results of the election)
- We can see that the most common word is "Paris", so we can guess that the answer is "Paris".
- This is roughly the approach that was used before LLMs.

How it works: LLMs and question answering

- LLMs are trained on a large corpus of text, which includes a lot of information about the world.
    - These are encoded in the weights of the model.
- When we ask a question, the LLM can use this information to generate an answer.
- The fine-tuning process makes the model **very eager** to provide an answer.

Problems with answering from the model's internal knowledge

- **Data cutoff**: the model only has access to information that was available at the time of training.
- The information might not be available in weights: maybe the model was not trained on the relevant information, or maybe it was trained on it but it is not encoded in a way that allows the model to retrieve it.
    - This is especially a problem for smaller models.
- **Hallucinations**: the model may generate an answer that is not based on any real information.

Answering from the context

- If the answer is provided in the input, the model can use this information to generate an answer.
- Much easier and more reliable than answering from the model's internal knowledge.
- We could, for instance, copy paste the whole Wikipedia page about France, and then ask the question "What is the capital of France?".
- Or, we can let the LLM do this for us.

Retrieval augmented generation (RAG)

- The model retrieves relevant information from an external source (e.g., a search engine, a knowledge graph, etc.) and then uses this information to generate an answer.
- This allows the model to access up-to-date information and to provide more accurate answers.
- This is the approach that is used by most modern question answering systems, including those based on LLMs.

How does RAG work?

- Generate a web query

```
Generate a web query that downloads the wikipedia article about France using wget
```

```
wget -O france.html https://en.wikipedia.org/wiki/France
```

What information should RAG use?

- Current top chatbots all use RAG to some extent.
- The most usual sources are common information available on the web:
  - Wikipedia
  - Google search results
  - News websites
- News websites are not particularly happy about this, because it means that the model is essentially stealing their content and using it to generate answers.
  - As a result, some news websites have started to block access to their content for these models.

Local RAG

- Instead of retrieving information from the web, we can also retrieve information from a local database or the internal website of a company.
- For instance a legal firm might have a database of their own prior cases
    - Confidential information
    - Not available on the web
- Creating such customized LLMs is becoming increasingly common, and it allows companies to leverage the power of LLMs while still maintaining control over their data and ensuring that the model has access to the relevant information.

Tool use

- We have seen that RAG boils down to:
  - the model generating a query
  - using some software to execute the query
  - loading the results back into the context
- We can use the same idea for a different purpose: we can use the model to generate a query that is executed by some software, and then load the results back into the context.

Tool use: doing calculations

- LLMs are not very good at doing calculations:

```
What is 123456789 * 987654321?
```

- Clearly, this was not in the training data. One could hope that the algorithm for multiplication was in the training data, but it is not clear that the model has learned this algorithm.
- Instead, we can use the same approach as RAG, but instead of retrieving information from the web, we can generate a short python script:

```
result = multi(123456789, 987654321)
print(result)
```

Tool use vs. performing reasoning by the LLM

- Humans are very rarely performing reasoning from first principles.
- Let us consider multiplication
  - We are all taught an algorithm for it in school, using pen and paper
  - Back in time, people also used to do multiplication using an abacus, which is a physical tool
  - Back in time people also learned various tricks for doing mental arithmetic - this is another tool
  - And of course, we can also use a calculator.
- We are not expected to invent these ourselves!
- The same is true for LLMs: we should not expect them to perform reasoning from first principles, but rather to use tools that allow them to perform the reasoning for us.

What kind of tools LLMs can use?

- Web retrieval (RAG)
- Calculators
- Code execution (e.g., python scripts)
- Optimizers (e.g., for solving optimization problems)
- Simulators (e.g., for simulating physical systems)
- Computer vision models (e.g., for analyzing images, Optical Character Recognition, etc.)

Pitfalls and dangers: hallucinations

- Hallucinations in LLM refers are answers that are generated by the model that are not based on any real information.
    - For instance: The next Star Wars movie will be called "Star Wars: Jedi Reborn" and will feature Timothée Chalamet as a dark Jedi.
- It is not the same thing as a wrong answer!
- Hallucinations are an effect of the **finetuning process**, which makes the model very eager to provide an answer, even when it does not have enough information to do so.
- Basically between:
    - An answer of a right **form**
    - An answer that "I don't know"
- The model chooses the first one.

Related problem: calibration

- Let us say that we ask the model a question, and it provides an answer.
- How confident should we be in this answer? We can ask the model to provide a confidence score for its answer, but how much should we trust this score?
- This is the problem of calibration: how well does the model's confidence in its answer match the actual accuracy of the answer?

Possible nightmare scenarios

- Fake legal cases in legal briefs (happened several times already)
- Invented science in scientific papers (also happened several times already)
- Customer service chatbots providing wrong information to customers
- Medical chatbots providing wrong information to patients (no major public cases yet, but it is only a matter of time)

Can we trust LLMs for question answering?

- It would be easy to say that "we should alway verify the answers provided by LLMs using a reliable source"
  - But this is not always possible, and very often negates the benefit of using an LLM in the first place.
- At minimum we should:
  - Be aware of the posibility of hallucinations
  - Understand the risk
- When possible, use RAG (or tool use more generally) to reduce the risk of hallucinations
- When possible, ask the model to provide evidence for its answers (eg. the webpage where it found the answer, the code it used to calculate the answer, etc.)

Try it out: prompt-based few-shot learning

Try it out: RAG