# Natural language processing: Text generation

# The problem: text generation

- Let us say we are applying for a job, and we want to write a cover letter.
- **Text generation**: create an appropriate text
- The objective is to generate a text that
  - **Fits the objective** of the user: i.e. it is indeed a cover letter
  - It is **contextually appropriate**: it is not just a cover letter, but one that is appropriate for the job we are applying for, and the applicant
  - It is **coherent**: it makes sense, and is not just a random collection of words
  - It is **well-written**: it is grammatically correct, and is not full of spelling mistakes

# Text generation

- **Inputs**:
  - Specification of the task in some format
  - Other information necessary (in some format)
- **Outputs**:
  - A text - normally in a plain text format
  - It can be in some cased rich text format (with formatting, itemization, sections etc.)
  - It can be in some structured format, e.g. a spreadsheet.
  - Quite often, choosing markdown as the output format is a good idea, as it allows for some formatting, but is still quite simple to generate.

# How do I know that it works?

- Often it is very difficult to evaluate the quality of the generated text, as it is often subjective.
- However, there are some objective measures that can be used, e.g.:
  - We can measure the number of spelling mistakes, or grammatical errors.
  - We can measure the coherence of the text, e.g. by checking if it contains certain keywords, or if it follows a certain structure.
- Human evaluation is often necessary, but it can be time-consuming and expensive.
- Ultimately, the best way to evaluate the quality of the generated text is to use it in a real-world application, and see if it achieves the desired outcome.
  - Does the cover letter lead to a job interview?

# Let us hack together a text generation system

- How would we create a text generation system with introductory programming skills?
- We can use a **template-based approach**: we can create a template for the text we want to generate, and then fill in the template with the appropriate information.
- For example, we can create a template for a cover letter, and then fill in the template with the applicant's information, and the job description.

# Template-based text generation

Dear [Hiring Manager],
I am writing to apply for the [Position] at [Company]. I believe that my skills and experience make me a strong candidate for this position. I have [X years] of experience in [Field], and I have worked on [Projects]. I am particularly interested in this position because [Reason]. I am confident that I can contribute to the success of [Company], and I would be excited to join your team. Thank you for considering my application. I look forward to the opportunity to discuss my qualifications further.
Sincerely, [Applicant]

# Template-based text generation: limitations

- The generated text is often very generic, and does not fit the specific context of the user
- It is hard to create templates for all possible scenarios, and to fill in the templates with the appropriate information
- It is hard to personalize it to the exact circumstances
- Nevertheless, it was and continues to be a very popular approach, and is still used in many applications, e.g. in chatbots, and in some email generation systems.

# Where does the information about the task and the context come from?

- In the template-based approach, the information about the task is in the template itself
- The information about the context usually comes from a database or other structured source
    - Company = "Microsoft"
    - Position = "Software Engineer"
    - Applicant = "John Doe"

# LLM based text generation

- The information about the task and the context is provided to the model in a **prompt**: a text that describes the task, and provides the necessary information.

# How does an LLM know how a cover letter should look like?

- The model has been trained on a large corpus of text, which includes many examples of cover letters, and other types of text.
- The model learns the structure and the style of the text, and can generate text that is similar to the examples it has seen during training.
- Alternatively (or in addition), the model can be provided with a few examples of the desired output, and can learn from these examples how to generate the appropriate text.
    - So the templates still have a role to play, but they are not hard-coded, but rather provided as examples in the prompt.

# How does the LLM know the context?

- The model can be provided with the necessary information in the prompt, e.g.:
  - "Write a cover letter for a software engineer position at Microsoft, for an applicant named John Doe."
- It is possible that the LLM operates with a system prompt that already contains some information about the user. For example, if the user has a profile in the system, the system prompt can contain information about the user's name, skills, experience etc., and the LLM can use this information to generate more personalized text.

# Application: write an official letter

- Make sure to prompt the LLM with all the necessary information:
  - Who is the recipient of the letter?
  - What is the purpose of the letter?
  - What is the context of the letter?
  - What is the desired tone of the letter?
- Make sure to provide examples of the desired output, if possible.
- Verify for accuracy, especially for hallucinations. Make sure that every information in the letter is actually coming from your own input.

# Try it out: The red door

- Prompt: Write a letter for the housing association about asking permission to paint my door red, which is not a color approved in the housing agreement.

# Benefits of LLM-based writing of official letters

- Save time and effort
- Democratises access to well-written letters, as not everyone has the same writing skills
- Helps the recipient of the letter, as it can be easier to read and understand.

# Pitfalls of LLM-based writing of official letters

- Looks generic, and looses the personal touch, which can be important in some contexts
  - Make sure that whatever makes your case special, is included in the prompt, and is reflected in the generated letter.
- The danger of hallucinations
- The danger of signing a letter that you did not write, which can have legal implications
  - Review the generated letter carefully.
  - You can use the LLM itself to help you with the understanding of the letter
    - For instance, by summarization
    - Asking to explain the letter in simple terms etc.
- Do not get surprised if you receive an LLM-generated answer.

# Application: write fiction in a certain style

- You can generate text in the style of a particular author, genre, or time period by providing examples in the prompt.
    - For widely known authors, the model has likely seen many examples of their writing during training so you don't need to provide examples.

# Example: The Matter of the Sulphur Cat

Create the summary of a novel in the style of Agatha Christie, which is about a murder that had been discovered due to the sulphuric smell of a cat.

# Pitfalls and dangers: issues of copyright and trademarks

- If the generated text is very similar to the original text, it can be considered a derivative work, and can be subject to copyright infringement.
    - Especially, if the generated text uses characters from the original work! - trademark infringement.
- It is unclear how the copyright law applies to the generated text, and it can depend on the jurisdiction.
- The reverse of this is that it might make it difficult to copyright the work that had been generated by the LLM