Transforming

In this notebook, we will explore how to use Large Language Models for text transformation tasks such as language translation, spelling and grammar checking, tone adjustment, and format conversion.

Large language models are very good at transforming its input to a different format, such as inputting a piece of text in one language and transforming it or translating it to a different language, or helping with spelling and grammar corrections.

So taking as input a piece of text that may not be fully grammatical and helping you to fix that up a bit, or even transforming formats, such as inputting HTML and outputting JSON. So there's a bunch of applications that I used to write somewhat painfully with a bunch of regular expressions that would definitely be much more simply implemented now with a large language model and a few prompts.

Yeah, I use ChatGPT to proofread pretty much everything I write these days, so I'm excited to show you some more examples in the notebook now.

Setup

So first we'll import OpenAI and also use the same get_completion helper function that we've been using throughout the lessons.

In [1]:
```
import openai
import os

from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv()) # read local .env file

openai.api_key  = os.getenv('OPENAI_API_KEY')
```

In [2]:
```
def get_completion(prompt, model="gpt-3.5-turbo", temperature=0):
    messages = [{"role": "user", "content": prompt}]
    response = openai.ChatCompletion.create(
        model=model,
        messages=messages,
        temperature=temperature,
    )
    return response.choices[0].message["content"]
```

Translation

The first thing we'll do is a translation task. So large language models are trained on a lot of text from many sources, a lot of which is the internet, and this is, of course, in many different languages. So this kind of imbues the model with the ability to do translation. And these models know hundreds of languages to varying degrees of proficiency. And so we'll go through some examples of how to use this capability.

`ChatGPT is trained with sources in many languages. This gives the model the ability to do translation. Here are some examples of how to use this capability.`

Example 1

So let's start off with something simple. So in this first example, the prompt is translate the following English text to Spanish. "Hi, I would like to order a blender". And the response is "Hola, me gustaría ordenar una licuadora".

In [4]:
```
prompt = f"""
Translate the following English text to Spanish: \
```Hi, I would like to order a blender```
"""
# response = get_completion(prompt)
# print(response)
```

Output:

```
prompt = f"""
Translate the following English text to Spanish: \
```Hi, I would like to order a blender```
"""
response = get_completion(prompt)
print(response)
```

```
Hola, me gustaría ordenar una licuadora.
```

Example 2

Okay, let's try another example. So in this example, the prompt is "Tell me what language this is". And then this is in French, "Combien coûte le lampadaire". And so let's run this. And the model has identified that "This is French."

In [5]:
```
prompt = f"""
Tell me which language this is:
```Combien coûte le lampadaire?```
"""
# response = get_completion(prompt)
# print(response)
```

Output:

```
prompt = f"""
Tell me which language this is:
```Combien coûte le lampadaire?```
"""
response = get_completion(prompt)
print(response)
```

```
This language is French.
```

Example 3

The model can also do multiple translations at once. So in this example, let's say translate the following text to French and Spanish. And you know what? Let's add another an English pirate. And the text is "I want to order a basketball".

In [6]:
```
prompt = f"""
Translate the following  text to French and Spanish
and English pirate: \
```I want to order a basketball```
"""
# response = get_completion(prompt)
# print(response)
```

Output:

```
prompt = f"""
Translate the following  text to French and Spanish
and English pirate: \
```I want to order a basketball```
"""
response = get_completion(prompt)
print(response)
```

```
French: ```Je veux commander un ballon de basket```
Spanish: ```Quiero ordenar una pelota de baloncesto```
English: ```I want to order a basketball```
```

So here we have French, Spanish and English pirate.

Example 4

So in some languages, the translation can change depending on the speaker's relationship to the listener. And you can also explain this to the language model. And so it will be able to translate accordingly. So in this example, we say, "Translate the following text to Spanish in both the formal and informal forms". "Would you like to order a pillow?" And also notice here we're using a different delimiter than these backticks. It doesn't really matter as long as there's kind of a clear separation.

In [ ]:
```
prompt = f"""
Translate the following text to Spanish in both the \
formal and informal forms:
'Would you like to order a pillow?'
"""
# response = get_completion(prompt)
# print(response)
```

Output:

```
prompt = f"""
Translate the following text to Spanish in both the \
formal and informal forms:
'Would you like to order a pillow?'
"""
response = get_completion(prompt)
print(response)
```

```
Formal: ¿Le gustaría ordenar una almohada?
Informal: ¿Te gustaría ordenar una almohada?
```

So here we have the formal and informal. So formal is when you're speaking to someone who's maybe senior to you or you're in a professional situation. That's when you use a formal tone and then informal is when you're speaking to maybe a group of friends.

Universal Translator

`Imagine you are in charge of IT at a large multinational e-commerce company. Users are messaging you with IT issues in all their native languages. Your staff is from all over the world and speaks only their native languages. You need a universal translator!`

So for the next example, we're going to pretend that we're in charge of a multinational e-commerce company and so the user messages are going to be in all different languages and so users are going to be telling us about their IT issues in a wide variety of different languages. So we need a universal translator. So first we'll just paste in a list of user messages in a variety of different languages.

In [7]:
```
user_messages = [
  "La performance du système est plus lente que d'habitude.",  # System performance is slower than normal
  "Mi monitor tiene píxeles que no se iluminan.",              # My monitor has pixels that are not lighting
  "Il mio mouse non funziona",                                 # My mouse is not working
  "Mój klawisz Ctrl jest zepsuty",                             # My keyboard has a broken control key
  "我的屏幕在闪烁"                                               # My screen is flashing
]
```

And now we will loop through each of these user messages. So "for issue in user_messages". And then I'm going to copy over this slightly longer code block. And so the first thing we'll do is ask the model to tell us what language the issue is in. So here's the prompt. Then we'll print out the original message's language and the issue. And then we'll ask the model to translate it into English and Korean.

In [10]:
```
for issue in user_messages:

    prompt = f"Tell me what language this is: ```{issue}```"

#     lang = get_completion(prompt)

#     print(f"Original message ({lang}): {issue}")

    prompt = f"""

    Translate the following  text to English \
    and Korean: ```{issue}```

    """
#     response = get_completion(prompt)
#     print(response, "\n")
```

So let's run this.

Output

```
user_messages = [
  "La performance du système est plus lente que d'habitude.",  # System performance is slower th
  "Mi monitor tiene píxeles que no se iluminan.",              # My monitor has pixels that are
  "Il mio mouse non funziona",                                 # My mouse is not working
  "Mój klawisz Ctrl jest zepsuty",                             # My keyboard has a broken contro
  "我的屏幕在闪烁"                                               # My screen is flashing
]
```

```
for issue in user_messages:
    prompt = f"Tell me what language this is: ```{issue}```"
    lang = get_completion(prompt)
    print(f"Original message ({lang}): {issue}")

    prompt = f"""
    Translate the following  text to English \
    and Korean: ```{issue}```
    """
    response = get_completion(prompt)
    print(response, "\n")
```

```
Original message (The language is French.): La performance du système est plus lente que d'habi
tude.
The performance of the system is slower than usual.

시스템의 성능이 평소보다 느립니다.

Original message (The language is Spanish.): Mi monitor tiene píxeles que no se iluminan.
English: "My monitor has pixels that do not light up."

Korean: "내 모니터에는 밝아지지 않는 픽셀이 있습니다."

Original message (The language is Italian.): Il mio mouse non funziona
English: "My mouse is not working."
Korean: "내 마우스가 작동하지 않습니다."

Original message (The language is Polish.): Mój klawisz Ctrl jest zepsuty
English: "My Ctrl key is broken"
Korean: "내 Ctrl 키가 고장 났어요"

Original message (The language is Chinese.): 我的屏幕在闪烁
English: My screen is flickering.
Korean: 내 화면이 깜박거립니다.
```

And so amazing, you've just built a universal translator. And also feel free to pause the video and add kind of any other languages you want to try here. Maybe languages you speak yourself and see how the model does.

Tone Transformation

`Writing can vary based on the intended audience. ChatGPT can produce different tones.`

So, the next thing we're going to dive into is tone transformation. Writing can vary based on an intended audience, you know, the way that I would write an email to a colleague or a professor is obviously going to be quite different to the way I text my younger brother. And so, ChatGPT can actually also help produce different tones. So, let's look at some examples.

Example

So, in this first example, the prompt is "Translate the following from slang to a business letter". "Dude, this is Joe, check out this spec on the standing lamp."

In [11]:
```
prompt = f"""
Translate the following from slang to a business letter:
'Dude, This is Joe, check out this spec on this standing lamp.'
"""
# response = get_completion(prompt)
# print(response)
```

So, let's execute this.

Output

```
prompt = f"""
Translate the following from slang to a business letter:
'Dude, This is Joe, check out this spec on this standing lamp.'
"""
response = get_completion(prompt)
print(response)
```

```
Dear Sir/Madam,

I hope this letter finds you well. My name is Joe, and I am writing to br
ing your attention to a specification document regarding a standing lamp.

I kindly request that you take a moment to review the attached document,
as it provides detailed information about the features and qualities of t
he aforementioned standing lamp.

Thank you for your time and consideration. I look forward to discussing t
his matter further with you.

Yours sincerely,
Joe
```

And as you can see, we have a much more formal business letter with a proposal for a standing lamp specification.

Format Conversion

`ChatGPT can translate between formats. The prompt should describe the input and output formats.`

The next thing that we're going to do is to convert between different formats. ChatGPT is very good at translating between different formats such as JSON to HTML, you know, XML, all kinds of things. Markdown. And so, in the prompt, we'll describe both the input and the output formats.

Example

Here is an example. So, we have this JSON that contains a list of restaurant employees with their name and email. And then in the prompt, we're going to ask the model to translate this from JSON to HTML. So, the prompt is "Translate the following Python dictionary from JSON to an HTML table with column headers and title". And then we'll get the response from the model and print it.

In [12]:
```
data_json = { "resturant employees" :[
    {"name":"Shyam", "email":"shyamjaiswal@gmail.com"},
    {"name":"Bob", "email":"bob32@gmail.com"},
    {"name":"Jai", "email":"jai87@gmail.com"}
]}

prompt = f"""
Translate the following python dictionary from JSON to an HTML \
table with column headers and title: {data_json}
"""
# response = get_completion(prompt)
# print(response)
```

So, here we have some HTML displaying all of the employee names and emails.

Output

```
data_json = { "resturant employees" :[
    {"name":"Shyam", "email":"shyamjaiswal@gmail.com"},
    {"name":"Bob", "email":"bob32@gmail.com"},
    {"name":"Jai", "email":"jai87@gmail.com"}
]}

prompt = f"""
Translate the following python dictionary from JSON to an HTML \
table with column headers and title: {data_json}
"""
response = get_completion(prompt)
print(response)
```

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
  font-family: arial, sans-serif;
  border-collapse: collapse;
  width: 100%;
}

td, th {
  border: 1px solid #dddddd;
  text-align: left;
  padding: 8px;
}

tr:nth-child(even) {
  background-color: #dddddd;
}
</style>
</head>
<body>

<h2>Restaurant Employees</h2>

<table>
  <tr>
    <th>Name</th>
    <th>Email</th>
  </tr>
  <tr>
    <td>Shyam</td>
    <td>shyamjaiswal@gmail.com</td>
  </tr>
  <tr>
    <td>Bob</td>
    <td>bob32@gmail.com</td>
  </tr>
  <tr>
    <td>Jai</td>
    <td>jai87@gmail.com</td>
  </tr>
</table>

</body>
</html>
```

And so, now let's see if we can actually view this HTML. So, we're going to use this display function from this Python library, "display (HTML(response))".

In [13]:
```
# from IPython.display import display, Markdown, Latex, HTML, JSON
# display(HTML(response))
```

And here you can see that this is a properly formatted HTML table. The next transformation task we're going to do is spell check and grammar checking. And this is a really kind of popular use for ChatGPT. I highly recommend doing this, I do this all the time. And it's especially useful when you're working in a non-native language. And so here are some examples of some common grammar and spelling problems and how the language model can help address these. So I'm going to paste in a list of sentences that have some grammatical or spelling errors.

Output

```
from IPython.display import display, Markdown, Latex, HTML, JSON
display(HTML(response))
```

Restaurant Employees

Name	Email
Shyam	shyamjaiswal@gmail.com
Bob	bob32@gmail.com
Jai	jai87@gmail.com

And here you can see that this is a properly formatted HTML table.

Spellcheck/Grammar check.

The next transformation task we're going to do is spell check and grammar checking. And this is a really kind of popular use for ChatGPT. I highly recommend doing this, I do this all the time. And it's especially useful when you're working in a non-native language.

Here are some examples of common grammar and spelling problems and the LLM's response.

Example 1

To signal to the LLM that you want it to proofread your text, you instruct the model to 'proofread' or 'proofread and correct'.

In [14]:
```
text = [
  "The girl with the black and white puppies have a ball.",  # The girl has a ball.
  "Yolanda has her notebook.", # ok
  "Its going to be a long day. Does the car need it's oil changed?",  # Homonyms
  "Their goes my freedom. There going to bring they're suitcases.",  # Homonyms
  "Your going to need you're notebook.",  # Homonyms
  "That medicine effects my ability to sleep. Have you heard of the butterfly affect?", # Homonyms
  "This phrase is to cherck chatGPT for speling abilitty"  # spelling
]
for t in text:
    prompt = f"""Proofread and correct the following text
    and rewrite the corrected version. If you don't find
    and errors, just say "No errors found". Don't use
    any punctuation around the text:
    ```{t}```"""
#     response = get_completion(prompt)
#     print(response)
```

Output

```
text = [
  "The girl with the black and white puppies have a ball.",  # The girl ha
  "Yolanda has her notebook.", # ok
  "Its going to be a long day. Does the car need it's oil changed?",  # Ho
  "Their goes my freedom. There going to bring they're suitcases.",  # Hom
  "Your going to need you're notebook.",  # Homonyms
  "That medicine effects my ability to sleep. Have you heard of the butter
  "This phrase is to cherck chatGPT for speling abilitty"  # spelling
]
for t in text:
    prompt = f"""Proofread and correct the following text
    and rewrite the corrected version. If you don't find
    and errors, just say "No errors found". Don't use
    any punctuation around the text:
    ```{t}```"""
    response = get_completion(prompt)
    print(response)
```

```
The girl with the black and white puppies has a ball.
No errors found.
No errors found.
There goes my freedom. They're going to bring their suitcases.
You're going to need your notebook.
That medicine affects my ability to sleep. Have you heard of the butterfl
y effect?
This phrase is to check chatGPT for spelling ability.
```

Example 2

And so now we'll do another example. It's always useful to check your text before you post it in a public forum. And so we'll go through an example of checking a review. And so here is a review about a stuffed panda. And so we're going to ask the model to proofread and correct the review.

In [15]:
```
text = f"""
Got this for my daughter for her birthday cuz she keeps taking \
mine from my room.  Yes, adults also like pandas too.  She takes \
it everywhere with her, and it's super soft and cute.  One of the \
ears is a bit lower than the other, and I don't think that was \
designed to be asymmetrical. It's a bit small for what I paid for it \
though. I think there might be other options that are bigger for \
the same price.  It arrived a day earlier than expected, so I got \
to play with it myself before I gave it to my daughter.
"""
prompt = f"proofread and correct this review: ```{text}```"


# response = get_completion(prompt)
# print(response)
```

Output

```
text = f"""
Got this for my daughter for her birthday cuz she keeps taking \
mine from my room.  Yes, adults also like pandas too.  She takes \
it everywhere with her, and it's super soft and cute.  One of the \
ears is a bit lower than the other, and I don't think that was \
designed to be asymmetrical. It's a bit small for what I paid for it \
though. I think there might be other options that are bigger for \
the same price.  It arrived a day earlier than expected, so I got \
to play with it myself before I gave it to my daughter.
"""
prompt = f"proofread and correct this review: ```{text}```"
response = get_completion(prompt)
print(response)
```

```
Got this for my daughter for her birthday because she keeps taking mine f
rom my room. Yes, adults also like pandas too. She takes it everywhere wi
th her, and it's super soft and cute. However, one of the ears is a bit l
ower than the other, and I don't think that was designed to be asymmetric
al. Additionally, it's a bit small for what I paid for it. I believe ther
e might be other options that are bigger for the same price. On the posit
ive side, it arrived a day earlier than expected, so I got to play with i
t myself before I gave it to my daughter.
```

Great. So we have this corrected version.

In [ ]:
```
# from redlines import Redlines

# diff = Redlines(text,response)
# display(Markdown(diff.output_markdown))
```

And one cool thing we can do is find the differences between our original review and the model's output. So we're going to use this redlines Python package to do this. And we're going to get the diff between the original text of our review and the model output and then display this.

```
from redlines import Redlines

diff = Redlines(text,response)
display(Markdown(diff.output_markdown))
```

Got this for my daughter for her birthday ~~cuz~~ because she keeps taking mine from my ~~room.~~ room. Yes, adults also like pandas ~~too.~~ too. She takes it everywhere with her, and it's super soft and ~~cute. One~~ cute. However, one of the ears is a bit lower than the other, and I don't think that was designed to be asymmetrical. ~~It's~~ Additionally, it's a bit small for what I paid for ~~it though.~~ it. I ~~think~~ believe there might be other options that are bigger for the same ~~price. It~~ price. On the positive side, it arrived a day earlier than expected, so I got to play with it myself before I gave it to my ~~daughter.~~ daughter.

And so here you can see the diff between the original review and the model output and the things that have been corrected.

Example 3

So, the prompt that we used was, "proofread and correct this review". But you can also make more dramatic changes, changes to tone, and that kind of thing. So, let's try one more thing. So, in this prompt, we're going to ask the model to proofread and correct this same review, but also make it more compelling and ensure that it follows APA style and targets an advanced reader. And we're also going to ask for the output in markdown format. And so we're using the same text from the original review up here.

In [16]:
```
prompt = f"""
proofread and correct this review. Make it more compelling.
Ensure it follows APA style guide and targets an advanced reader.
Output in markdown format.
Text: ```{text}```
"""
# response = get_completion(prompt)
# display(Markdown(response))
```

So, let's execute this.

Output

```
prompt = f"""
proofread and correct this review. Make it more compelling.
Ensure it follows APA style guide and targets an advanced reader.
Output in markdown format.
Text: ```{text}```
"""
response = get_completion(prompt)
display(Markdown(response))
```

Review of a Panda Plush Toy: A Perfect Gift for All Ages

I purchased this adorable panda plush toy as a birthday gift for my daughter, who has a penchant for sneaking into my room and taking mine. It's worth noting that pandas are not just for kids; even adults can't resist their charm.

From the moment my daughter unwrapped this gift, she has been inseparable from it. The plush toy's irresistibly soft and cuddly texture makes it the perfect companion for her daily adventures. However, I did notice a minor flaw in its design - one of the ears is slightly lower than the other, which seems unintentional. Nevertheless, this asymmetry does not detract from its overall cuteness.

While I am thoroughly satisfied with the quality and appeal of this panda plush toy, I must admit that I expected it to be slightly larger considering its price. It would be wise to explore other options that offer a larger size for the same price range.

On a positive note, the delivery of this product exceeded my expectations. It arrived a day earlier than anticipated, allowing me to indulge in some playtime with the plush toy before presenting it to my daughter. This unexpected bonus added to the overall excitement and joy surrounding this purchase.

In conclusion, this panda plush toy is a delightful gift that transcends age boundaries. Its exceptional softness and undeniable cuteness make it an irresistible companion for anyone. While the size may not meet everyone's expectations, the prompt delivery and the sheer joy it brings to its recipient make it a worthwhile purchase.

And here we have an expanded APA style review of the softpanda.

Try it yourself!

Try changing the instructions to form your own review.

Thanks to the following sites:

https://writingprompts.com/bad-grammar-examples/