Project Name: CSML1010 NLP Course Project - Part 1 - Proposal): Problem, Dataset, and Exploratory Data Analysis

Authors (Group3): Paul Doucet, Jerry Khidaroo

Project Repository: https://github.com/CSML1010-3-2020/NLPCourseProject (https://github.com/CSML1010-3-2020/NLPCourseProject)

1. Problem Definition and Data Preparation Notebook:

This notebook will review the following sections as part of our project proposal:

Problem Definition

The problem we will be analysing is supervised text classification. The goal is to investigate which supervised mahine learning methods will give the best results in classifying the texts from our dataset into the pre-defined categories. This is a multi-class text classification problem. The input will be the text elements of each conversation concatenated together. The output will be the instruction_id.

Dataset Description

The dataset we will be using for our project is the **Taskmaster-1** dataset from Google. Taskmaster-1 (https://research.google/tools/datasets/taskmaster-1/)

The dataset can be obtained from: https://github.com/google-research-datasets/Taskmaster (https://github.com/google-research-datasets/Taskmaster)

> The dataset consists of 13,215 task-based dialogs, including 5,507 spoken and 7,708 written dialogs created with two distinct procedures. Each conversation falls into one of six domains: ordering pizza, creating auto repair appointments, setting up ride service, ordering movie tickets, ordering coffee drinks and making restaurant reservations. Our initial data exploration will use the written dialog file with 7,708 records.

Roadmap

As part of our study, we will be consider the following steps to find the ideal classifier for incoming texts.

- **Feature Engineering:**
  - Count Vectors: Count Vector is a matrix notation of the dataset in which every row represents a document from the corpus, every column represents a term from the corpus, and every cell represents the frequency count of a particular term in a particular document. These provide no context, nor any consideration of the words in relation to other words or position in the sentence.
    - Bag-of-words
    - Bag of n-grams
  - TF-IDF Vectors: TF-IDF score represents the relative importance of a term in the document and the entire corpus. TF-IDF score is composed by two terms: the first computes the normalized Term Frequency (TF), the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.
    - Word level Tfidf
    - N-gram Level TF-IDF
  - Word Embeddings: A word embedding is a form of representing words and documents using a dense vector representation. The position of a word within the vector space is learned from text and is based on the words that surround the word when it is used. These generate a context free representation of each word in the vocabulary.
    - Word2vec
    - Glove
  - NLP Based features: An example of this would be Frequency distribution of Part of Speech Tags.
    - Noun, Verb, Adjective, Adverb, Pronoun Counts
  - Language Models: These are recent breakthroughs that provide context and generate a representation of each word based on other words in the sentence.
    - BERT or FLAIR
- **Model Training:**
  - Naive Bayes (multinominal): the one most suitable for word counts is multinominal.
  - logistic regression.
  - support vector machine.
  - decision tree (random forest).
  - Ensemble: Bagging, Boosting
- **Model Evaluation:**
  - Confusion Matrix
  - Metrics: Presicion, Recall, F1 Score

Import the Dataset

Two JSON format file we will be using from the **Taskmaster-1** dataset is the following:

- **self-dialogs.json** contains all the one-person dialogs.

This file can be divided into train/dev/test sets by matching the dialog IDs from the following files:

- train.csv
- dev.csv
- test.csv

Supplementary information is provided to describe the data structure and annotation schema.

- **sample.json** - A sample conversation describing the format of the data.
- **ontology.json** - Schema file describing the annotation ontology.

The structure of the conversations in the data files is as follows:

- **conversationId:** A universally unique identifier with the prefix 'dlg-'. The ID has no meaning.
- **utterances:** An array of utterances that make up the conversation.
- **instructionId:** A reference to the file(s) containing the user (and, if applicable, agent) instructions for this conversation.

The **utterances** category, has the following sub-categories of which we will be using the **text** to perform our analysis:

- **index:** A 0-based index indicating the order of the utterances in the conversation.
- **speaker:** Either USER or ASSISTANT, indicating which role generated this utterance.
- **text:** The raw text of the utterance. In case of self dialogs, this is written by the crowdsourced worker. In case of the WOz dialogs, 'ASSISTANT' turns are written and 'USER' turns are transcribed from the spoken recordings of crowdsourced workers.
- **segments:** An array of various text spans with semantic annotations.

Import the libraries

In [1]:
```
import json
import pandas as pd
from pandas.io.json import json_normalize
```

Open the `self-dialogs.json` file and view the entire content

In [2]:
```
with open(r'./data/self-dialogs.json') as f:
    data = json.load(f)
```

Extract the `utterances` column and normalize it to view all individual text fields.
This will increase the dataframe rows from 7708 to 169469 as each text field is now available

In [3]:
```
tt = pd.json_normalize(data, 'utterances', ['conversation_id','instruction_id'])
```

View the dataframe with the text field visible outside the dictionary

In [4]:
```
tt
```

Out[4]:

	index	speaker	text	segments	conversation_id	instruction_id
0	0	USER	Hi, I'm looking to book a table for Korean fod.	NaN	dlg-00055f4e-4a46-48bf-8d99-4e477663eb23	restaurant-table-2
1	1	ASSISTANT	Ok, what area are you thinking about?	NaN	dlg-00055f4e-4a46-48bf-8d99-4e477663eb23	restaurant-table-2
2	2	USER	Somewhere in Southern NYC, maybe the East Vill...	[{'start_index': 13, 'end_index': 49, 'text': ...	dlg-00055f4e-4a46-48bf-8d99-4e477663eb23	restaurant-table-2
3	3	ASSISTANT	Ok, great. There's Thursday Kitchen, it has g...	[{'start_index': 20, 'end_index': 35, 'text': ...	dlg-00055f4e-4a46-48bf-8d99-4e477663eb23	restaurant-table-2
4	4	USER	That's great. So I need a table for tonight at...	[{'start_index': 26, 'end_index': 31, 'text': ...	dlg-00055f4e-4a46-48bf-8d99-4e477663eb23	restaurant-table-2
...	...	...	...	...	...	...
169464	15	ASSISTANT	Ok.	NaN	dlg-fffa6565-32bb-4592-8d30-fff66df29633	movie-tickets-3
169465	16	USER	I think we'll pass for tonight. Thanks anyhow.	NaN	dlg-fffa6565-32bb-4592-8d30-fff66df29633	movie-tickets-3
169466	17	ASSISTANT	Ok. Just let me know if you change your mind.	NaN	dlg-fffa6565-32bb-4592-8d30-fff66df29633	movie-tickets-3
169467	18	USER	I will. Thanks	NaN	dlg-fffa6565-32bb-4592-8d30-fff66df29633	movie-tickets-3
169468	19	ASSISTANT	No problem!	NaN	dlg-fffa6565-32bb-4592-8d30-fff66df29633	movie-tickets-3

169469 rows × 6 columns

Remove all columns but the `text` and `conversation_id` from the dataframe and view

tt.drop('index', axis=1, inplace=True) tt.drop('segments', axis=1, inplace=True) tt.drop('speaker', axis=1, inplace=True) tt

View the columns of the dataframe

In [5]:
```
tt.columns
```

Out[5]:
```
Index(['index', 'speaker', 'text', 'segments', 'conversation_id',
       'instruction_id'],
      dtype='object')
```

View the content of the `text` column, then the `conversation_id`

In [6]:
```
tt['text']
```

Out[6]:
```
0           Hi, I'm looking to book a table for Korean fod.
1                     Ok, what area are you thinking about?
2         Somewhere in Southern NYC, maybe the East Vill...
3         Ok, great.  There's Thursday Kitchen, it has g...
4         That's great. So I need a table for tonight at...
                              ...
169464                                                  Ok.
169465       I think we'll pass for tonight. Thanks anyhow.
169466        Ok. Just let me know if you change your mind.
169467                                        I will. Thanks
169468                                           No problem!
Name: text, Length: 169469, dtype: object
```

In [7]:
```
tt['conversation_id']
```

Out[7]:
```
0         dlg-00055f4e-4a46-48bf-8d99-4e477663eb23
1         dlg-00055f4e-4a46-48bf-8d99-4e477663eb23
2         dlg-00055f4e-4a46-48bf-8d99-4e477663eb23
3         dlg-00055f4e-4a46-48bf-8d99-4e477663eb23
4         dlg-00055f4e-4a46-48bf-8d99-4e477663eb23
                            ...
169464    dlg-fffa6565-32bb-4592-8d30-fff66df29633
169465    dlg-fffa6565-32bb-4592-8d30-fff66df29633
169466    dlg-fffa6565-32bb-4592-8d30-fff66df29633
169467    dlg-fffa6565-32bb-4592-8d30-fff66df29633
169468    dlg-fffa6565-32bb-4592-8d30-fff66df29633
Name: conversation_id, Length: 169469, dtype: object
```

View of one line of the dataframe filtered by `conversation_id`

In [8]:
```
tt[tt.conversation_id == 'dlg-00055f4e-4a46-48bf-8d99-4e477663eb23']
```

			you thinking about?		4e477663eb23	
2	2	USER	Somewhere in Southern NYC, maybe the East Vill...	[{'start_index': 13, 'end_index': 49, 'text': ...	dlg-00055f4e-4a46-48bf-8d99-4e477663eb23	restaurant-table-2
3	3	ASSISTANT	Ok, great. There's Thursday Kitchen, it has g...	[{'start_index': 20, 'end_index': 35, 'text': ...	dlg-00055f4e-4a46-48bf-8d99-4e477663eb23	restaurant-table-2
4	4	USER	That's great. So I need a table for tonight at...	[{'start_index': 26, 'end_index': 31, 'text': ...	dlg-00055f4e-4a46-48bf-8d99-4e477663eb23	restaurant-table-2
5	5	ASSISTANT	They don't have any availability for 7 pm.	[{'start_index': 37, 'end_index': 41, 'text': ...	dlg-00055f4e-4a46-48bf-8d99-4e477663eb23	restaurant-table-2
6	6	USER	What times are available?	NaN	dlg-00055f4e-4a46-48bf-8d99-4e477663eb23	restaurant-table-2
7	7	ASSISTANT	5 or 8.	[{'start_index': 0, 'end_index': 1,	dlg-00055f4e-4a46-48bf-8d99-	restaurant-

Categorize the `conversation_id` TODO: confirm this step is necessary

In [9]:
```
tt2 = tt.conversation_id.unique()
```

In [10]:
```
tt2
```

Out[10]:
```
array(['dlg-00055f4e-4a46-48bf-8d99-4e477663eb23',
       'dlg-0009352b-de51-474b-9f13-a2b0b2481546',
       'dlg-00123c7b-15a0-4f21-9002-a2509149ee2d', ...,
       'dlg-ffcd1d53-c080-4acf-897d-48236513bc58',
       'dlg-ffd9db94-36e3-4534-b99d-89f7560db17c',
       'dlg-fffa6565-32bb-4592-8d30-fff66df29633'], dtype=object)
```

Verify the length of the `tt2` array to confirm the number of conversations: note that it should match initial dataframe length of 7708

In [11]:
```
len(tt2)
```

Out[11]:
```
7708
```

Loop thru the entire `tt2` dataframe and combine all the text based on the conversation_id

In [12]:
```
# Loop thru all the conversation_id unique values
#df = pd.DataFrame(columns=['Conversation', 'ident'])
conversation_id = []
conv_text = []
instr_id = []
for i in tt2:
    conv2 = ''
    tti = tt[tt.conversation_id == i]
    conv = ''
    conv2 = ''
    instr3 = tti['instruction_id']
    instr_id.append(instr3.iloc[0])
    for j in tti:
        conv = tti['text']
    for k in conv:
        conv2 = conv2 + k + " "
    conversation_id.append(i)
    conv_text.append(conv2)
```

In [13]:
```
# View the content of the concatenated conversation list, created by combining al
conv_text[0:5]
```

Out[13]:
```
["Hi, I'm looking to book a table for Korean fod. Ok, what area are you think
ing about? Somewhere in Southern NYC, maybe the East Village? Ok, great.  The
re's Thursday Kitchen, it has great reviews. That's great. So I need a table
for tonight at 7 pm for 8 people. We don't want to sit at the bar, but anywhe
re else is fine. They don't have any availability for 7 pm. What times are av
ailable? 5 or 8. Yikes, we can't do those times. Ok, do you have a second cho
ice? Let me check. Ok. Lets try Boka, are they free for 8 people at 7? Yes. G
reat, let's book that. Ok great, are there any other requests? No, that's it,
just book. Great, should I use your account you have open with them? Yes plea
se. Great. You will get a confirmation to your phone soon. ",
 "Hi I would like to see if the Movie What Men Want is playing here. Yes it's
showing here would you like to purchase a ticket? Yes, for me and a friend so
two tickets please Okay. What time is that moving playing today? That movie i
s showing at 4, 5, and 8pm. Okay. Is there anymore movies showing around 8pm
Yes , showing at 8pm is Green Book. What is that about? It's about two men de
aling with racisim. Oh, no can you recommend anything else? What do you like?
Well I like movies that are funny. Like comedies? Well no I like action as we
ll. Okay. How to train your dragon is playing at 8pm. Okay can i get two tick
ets for that ? So you want me to cancel the tickets for What men want ? Yes p
```

In [14]:
```
# View the content of the conversation_id list, which will be used to merge with
conversation_id[0:5]
```

Out[14]:
```
['dlg-00055f4e-4a46-48bf-8d99-4e477663eb23',
 'dlg-0009352b-de51-474b-9f13-a2b0b2481546',
 'dlg-00123c7b-15a0-4f21-9002-a2509149ee2d',
 'dlg-0013673c-31c6-4565-8fac-810e173a5c53',
 'dlg-001d8bb1-6f25-4ecd-986a-b7eeb5fa4e19']
```

In [15]:
```
instr_id[0:5]
```

Out[15]:
```
['restaurant-table-2',
 'movie-tickets-1',
 'movie-tickets-3',
 'pizza-ordering-2',
 'pizza-ordering-2']
```

In [16]:
```
# Create a dictionary to store the conversation_id and text lists, which will be
ex_dict = {'id':conversation_id, 'conv':conv_text, 'instr':instr_id}
```

In [17]:
```
# Create a dataframe with the conversation id and conversation
df = pd.DataFrame(ex_dict)
df.columns = ['id', 'Conversation','Instruction_id']
df
```

Out[17]:

	id	Conversation	Instruction_id
0	dlg-00055f4e-4a46-48bf-8d99-4e477663eb23	Hi, I'm looking to book a table for Korean fod...	restaurant-table-2
1	dlg-0009352b-de51-474b-9f13-a2b0b2481546	Hi I would like to see if the Movie What Men W...	movie-tickets-1
2	dlg-00123c7b-15a0-4f21-9002-a2509149ee2d	I want to watch avengers endgame where do you ...	movie-tickets-3
3	dlg-0013673c-31c6-4565-8fac-810e173a5c53	I want to order a pizza from Bertuccis in Chel...	pizza-ordering-2
4	dlg-001d8bb1-6f25-4ecd-986a-b7eeb5fa4e19	Hi I'd like to order two large pizzas. Sure, w...	pizza-ordering-2
...	...	...	...
7703	dlg-ffc0c5fb-573f-40e0-b739-0e55d84100e8	I feel like eating at a nice restaurant tonigh...	restaurant-table-1
7704	dlg-ffc87550-389a-432e-927e-9a9438fc4f1f	Hi Sally, I need a Grande iced Americano with ...	coffee-ordering-2
7705	dlg-ffcd1d53-c080-4acf-897d-48236513bc58	Good afternoon. I would like to order a pizza ...	pizza-ordering-2
7706	dlg-ffd9db94-36e3-4534-b99d-89f7560db17c	Hey. I'm thinking of seeing What Men Want toni...	movie-tickets-1
7707	dlg-fffa6565-32bb-4592-8d30-fff66df29633	Hello. Can you help me purchase a couple of mo...	movie-tickets-3

7708 rows × 3 columns

In [18]:
```
# View first three rows of the data frame conversation columns
df['Conversation'][0:3]
```

Out[18]:
```
0    Hi, I'm looking to book a table for Korean fod...
1    Hi I would like to see if the Movie What Men W...
2    I want to watch avengers endgame where do you ...
Name: Conversation, dtype: object
```

In [19]:
```
# Export the dataframe to csv to confirm content
df.to_csv(r'./data/DF_selfDialogs.csv', index=False)
```

In [ ]:
```

```

Project Name: CSML1010 NLP Course Project - Part 1 - Proposal): Problem, Dataset, and Exploratory Data Analysis

Authors (Group3): Paul Doucet, Jerry Khidaroo

2. Data Clean-up and NLP Notebook

This notebook will review the Data Cleaning tasks performed as part of our project proposal:

Categorize Groups

In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv("./data/DF_selfDialogs.csv")
```

Import CSV

In [3]:
```
print (df.groupby('Instruction_id').size())
```

```
Instruction_id
auto-repair-appt-1    1161
coffee-ordering-1      735
coffee-ordering-2      641
movie-finder            54
movie-ticket-1          37
movie-tickets-1        642
movie-tickets-2        377
movie-tickets-3        195
pizza-ordering-1       257
pizza-ordering-2      1211
restaurant-table-1     704
restaurant-table-2     494
restaurant-table-3     102
uber-lyft-1            646
uber-lyft-2            452
dtype: int64
```

We need to fix the 37 movie-ticket-1 instruction_ids

In [4]:
```
df = df.replace(['movie-ticket-1'], 'movie-tickets-1')
```

In [5]:
```
print (df.groupby('Instruction_id').size())
```

```
Instruction_id
auto-repair-appt-1    1161
coffee-ordering-1      735
coffee-ordering-2      641
movie-finder            54
movie-tickets-1        679
movie-tickets-2        377
movie-tickets-3        195
pizza-ordering-1       257
pizza-ordering-2      1211
restaurant-table-1     704
restaurant-table-2     494
restaurant-table-3     102
uber-lyft-1            646
uber-lyft-2            452
dtype: int64
```

Add the Service Type as a column (i.e. auto, coffee, movie, etc.)

In [6]:
```
df['service_type'] = df['Instruction_id'].str.split('-',expand=True)[0]
print (df.groupby('service_type').size())
```

```
service_type
auto          1161
coffee        1376
movie         1305
pizza         1468
restaurant    1300
uber          1098
dtype: int64
```

In [7]:
```
df
```

Out[7]:

	id	Conversation	Instruction_id	service_type
0	dlg-00055f4e-4a46-48bf-8d99-4e477663eb23	Hi, I'm looking to book a table for Korean fod...	restaurant-table-2	restaurant
1	dlg-0009352b-de51-474b-9f13-a2b0b2481546	Hi I would like to see if the Movie What Men W...	movie-tickets-1	movie
2	dlg-00123c7b-15a0-4f21-9002-a2509149ee2d	I want to watch avengers endgame where do you ...	movie-tickets-3	movie
3	dlg-0013673c-31c6-4565-8fac-810e173a5c53	I want to order a pizza from Bertuccis in Chel...	pizza-ordering-2	pizza
4	dlg-001d8bb1-6f25-4ecd-986a-b7eeb5fa4e19	Hi I'd like to order two large pizzas. Sure, w...	pizza-ordering-2	pizza
...	...	...	...	...
7703	dlg-ffc0c5fb-573f-40e0-b739-0e55d84100e8	I feel like eating at a nice restaurant tonigh...	restaurant-table-1	restaurant
7704	dlg-ffc87550-389a-432e-927e-9a9438fc4f1f	Hi Sally, I need a Grande iced Americano with ...	coffee-ordering-2	coffee
7705	dlg-ffcd1d53-c080-4acf-897d-48236513bc58	Good afternoon. I would like to order a pizza ...	pizza-ordering-2	pizza
7706	dlg-ffd9db94-36e3-4534-b99d-89f7560db17c	Hey. I'm thinking of seeing What Men Want toni...	movie-tickets-1	movie
7707	dlg-fffa6565-32bb-4592-8d30-fff66df29633	Hello. Can you help me purchase a couple of mo...	movie-tickets-3	movie

7708 rows × 4 columns

Connect to Database

In [8]:
```
import sqlite3
con = sqlite3.connect('selfdialogs.db')
```

Cleaning the Dataset for NLP

Cleaning Function

In [9]:
```
import re
def clean(s):
    s = s.replace(r'<lb>', "\n")
    s = s.replace(r'<tab>', "\i")
    s = re.sub(r'<br */*>', "\n", s)
    s = s.replace("&lt;", "<").replace("&gt;", ">").replace("&amp;", "&")
    s = s.replace("&amp;", "&")
    # markdown urls
    s = re.sub(r'\(https*://[^\)]*\)', "", s)
    # normal urls
    s = re.sub(r'https*://[^\s]*', "", s)
    s = re.sub(r'_+', ' ', s)
    s = re.sub(r'"+', '"', s)
    return str(s)
```

In [10]:
```
df["selfdialog_clean"] = ''
```

Iterate and Clean

In [11]:
```
for i, row in df.iterrows():
    df.at[i, "selfdialog_clean"] = clean(row.Conversation)
```

In [12]:
```
df.head()
```

Out[12]:

	id	Conversation	Instruction_id	service_type	selfdialog_clean
0	dlg-00055f4e-4a46-48bf-8d99-4e477663eb23	Hi, I'm looking to book a table for Korean fod...	restaurant-table-2	restaurant	Hi, I'm looking to book a table for Korean fod...
1	dlg-0009352b-de51-474b-9f13-a2b0b2481546	Hi I would like to see if the Movie What Men W...	movie-tickets-1	movie	Hi I would like to see if the Movie What Men W...
2	dlg-00123c7b-15a0-4f21-9002-a2509149ee2d	I want to watch avengers endgame where do you ...	movie-tickets-3	movie	I want to watch avengers endgame where do you ...
3	dlg-0013673c-31c6-4565-8fac-810e173a5c53	I want to order a pizza from Bertuccis in Chel...	pizza-ordering-2	pizza	I want to order a pizza from Bertuccis in Chel...
4	dlg-001d8bb1-6f25-4ecd-986a-b7eeb5fa4e19	Hi I'd like to order two large pizzas. Sure, w...	pizza-ordering-2	pizza	Hi I'd like to order two large pizzas. Sure, w...

NLP

In [13]:
```
import spacy
nlp = spacy.load('en')
```

Iterate and Perform NLP

In [14]:
```
for i, row in df.iterrows():
    if i % 1000 == 0:
        print(i)
    if(row["selfdialog_clean"] and len(str(row["selfdialog_clean"])) < 1000000):
        doc = nlp(str(row["selfdialog_clean"]))
        adjectives = []
        nouns = []
        verbs = []
        lemmas = []

        for token in doc:
            lemmas.append(token.lemma_)
            if token.pos_ == "ADJ":
                adjectives.append(token.lemma_)
            if token.pos_ == "NOUN" or token.pos_ == "PROPN":
                nouns.append(token.lemma_)
            if token.pos_ == "VERB":
                verbs.append(token.lemma_)

        df.at[i, "selfdialog_lemma"] = " ".join(lemmas)
        df.at[i, "selfdialog_nouns"] = " ".join(nouns)
        df.at[i, "selfdialog_adjectives"] = " ".join(adjectives)
        df.at[i, "selfdialog_verbs"] = " ".join(verbs)
        df.at[i, "selfdialog_nav"] = " ".join(nouns+adjectives+verbs)
        df.at[i, "no_tokens"] = len(lemmas)
```

```
0
1000
2000
3000
4000
5000
6000
7000
```

In [15]:
```
df.head()
```

Out[15]:

	id	Conversation	Instruction_id	service_type	selfdialog_clean	selfdialog_lemma	sel
0	dlg-00055f4e-4a46-48bf-8d99-4e477663eb23	Hi, I'm looking to book a table for Korean fod...	restaurant-table-2	restaurant	Hi, I'm looking to book a table for Korean fod...	hi , -PRON- be look to book a table for korean...	so
1	dlg-0009352b-de51-474b-9f13-a2b0b2481546	Hi I would like to see if the Movie What Men W...	movie-tickets-1	movie	Hi I would like to see if the Movie What Men W...	hi -PRON- would like to see if the movie what ...	n tic
2	dlg-00123c7b-15a0-4f21-9002-a2509149ee2d	I want to watch avengers endgame where do you ...	movie-tickets-3	movie	I want to watch avengers endgame where do you ...	-PRON- want to watch avenger endgame where do ...	ave tim
3	dlg-0013673c-31c6-4565-8fac-810e173a5c53	I want to order a pizza from Bertuccis in Chel...	pizza-ordering-2	pizza	I want to order a pizza from Bertuccis in Chel...	-PRON- want to order a pizza from bertuccis in...	wh
4	dlg-001d8bb1-6f25-4ecd-986a-b7eeb5fa4e19	Hi I'd like to order two large pizzas. Sure, w...	pizza-ordering-2	pizza	Hi I'd like to order two large pizzas. Sure, w...	hi -PRON- would like to order two large pizza ...	

Store to Database

In [16]:
```
df.to_sql('posts_nlp', con, if_exists='replace')
```

In [ ]:
```
```

Project Name: CSML1010 NLP Course Project - Part 1 - Proposal): Problem, Dataset, and Exploratory Data Analysis

Authors (Group3): Paul Doucet, Jerry Khidaroo

3. Data Exploration Notebook

This notebook will review the Data Exploration tasks performed as part of our project proposal:

Load the Dataframe

In [1]:
```
# filter warnings on depreciation etc.
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
# import pandas, numpy
import pandas as pd
import numpy as np

# adjust pandas display
pd.options.display.max_columns = 30
pd.options.display.max_rows = 100
pd.options.display.float_format = '{:.2f}'.format
pd.options.display.precision = 2
pd.options.display.max_colwidth = -1
```

In [3]:
```
# Import matplotlib and seaborn and adjust some defaults
%matplotlib inline
%config InlineBackend.figure_format = 'svg'

from matplotlib import pyplot as plt
plt.rcParams['figure.dpi'] = 100

import seaborn as sns
sns.set_style("whitegrid")
```

Load Data

In [4]:
```
import sqlite3

sql = """
SELECT p.*
FROM posts_nlp p
"""

with sqlite3.connect('selfdialogs.db') as con:
    df = pd.read_sql_query(sql, con)
```

Data Exploration

In [5]:
```
# list column names and datatypes
df.dtypes
```

Out[5]:
```
index                 int64
id                    object
Conversation          object
Instruction_id        object
service_type          object
selfdialog_clean      object
selfdialog_lemma      object
selfdialog_nouns      object
selfdialog_adjectives object
selfdialog_verbs      object
selfdialog_nav        object
no_tokens             float64
dtype: object
```

In [6]:
```
# select a sample of some data frame columns
df[['id', 'Conversation', 'Instruction_id','service_type']] \
  .sample(2, random_state=42)
```

Out[6]:

	id	C
6482	dlg-d601e9c1-f9b4-4778-ae20-29f5ab6edd72	Hello can you please book a reservation at the crawling crab for tonight at 7:30 for 3 peop table available at that time how about 8:00 pm Ok That table is outside is that ok No that wo there another restaurant that you would like to try Yes how about the crying tree Same Tir Yes Ok I will book the table would you like the restaurant to send u a text confirmatior phone number 867 5309 Ok they will text u around 15 minutes before the table is ready table Ok Do you want to order drinks to have ready when you arrive. Yes please order 2 wine Ok they will have your drinks ready Ok To confirm I have a table for 3 ready at the today at 7:30 wine will be ordered prior to arriving they will text to confirm is this correc
6872	dlg-e3797e80-a033-47d3-be9f-3235ea00f09c	Hi, I would like for you to order a car for me From where would you like to leave? Ba Stadium, South Mint Street, Charlotte, NC And where will you be going? Romare Bearde Church Street, Charlotte, NC $6.65 Is Uber XL available ?Uber XL is available for 7.75. Are there any other options ?Black is available for 15.00 and Black SUV is available for 25.00 I would like to book U . You would like to book Uber XL for 7.75 ?Yes, that's correct . Ok I am booking your Uber XL now. Thank you. Do you have any other requests ?How much was the total in the end? It was When can I expect my UberXL to arrive? Your ride is on the way and you can check your s ph

In [7]:
```
# length of a dataframe
len(df)
```

Out[7]:
```
7708
```

In [8]:
```
# number of values per column
df.count()
```

Out[8]:
```
index                   7708
id                      7708
Conversation            7708
Instruction_id          7708
service_type            7708
selfdialog_clean        7708
selfdialog_lemma        7708
selfdialog_nouns        7708
selfdialog_adjectives   7708
selfdialog_verbs        7708
selfdialog_nav          7708
no_tokens               7708
dtype: int64
```

In [9]:
```
# size info, including memory consumption
df.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7708 entries, 0 to 7707
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   index                  7708 non-null   int64
 1   id                     7708 non-null   object
 2   Conversation           7708 non-null   object
 3   Instruction_id         7708 non-null   object
 4   service_type           7708 non-null   object
 5   selfdialog_clean       7708 non-null   object
 6   selfdialog_lemma       7708 non-null   object
 7   selfdialog_nouns       7708 non-null   object
 8   selfdialog_adjectives  7708 non-null   object
 9   selfdialog_verbs       7708 non-null   object
 10  selfdialog_nav         7708 non-null   object
 11  no_tokens              7708 non-null   float64
dtypes: float64(1), int64(1), object(10)
memory usage: 36.8 MB
```

Column Exploration

In [10]:
```
columns = [col for col in df.columns if not col.startswith('self')]
columns
```

Out[10]:
```
['index', 'id', 'Conversation', 'Instruction_id', 'service_type', 'no_tokens']
```

In [11]:
```
# describe categorical columns of type np.object
df[['service_type','Instruction_id']] \
  .describe(include=np.object) \
  .transpose()
```

Out[11]:

	count	unique	top	freq
service_type	7708	6	pizza	1468
Instruction_id	7708	14	pizza-ordering-2	1211

In [12]:
```
df['Instruction_id'].value_counts()[:10]
```

Out[12]:
```
pizza-ordering-2      1211
auto-repair-appt-1    1161
coffee-ordering-1     735
restaurant-table-1    704
movie-tickets-1       679
uber-lyft-1           646
coffee-ordering-2     641
restaurant-table-2    494
uber-lyft-2           452
movie-tickets-2       377
Name: Instruction_id, dtype: int64
```

In [13]:
```
# describe numerical columns
df.describe().transpose()
```

Out[13]:

	count	mean	std	min	25%	50%	75%	max
index	7708.00	3853.50	2225.25	0.00	1926.75	3853.50	5780.25	7707.00
no_tokens	7708.00	228.51	80.57	20.00	175.00	215.00	267.00	1336.00

In [14]:
```
# group by service_type, count distinct Instruction_id
cat_df = df.groupby('service_type') \
           .agg({'Instruction_id': pd.Series.nunique,
                 'id': pd.Series.count}) \
           .rename(columns={'Instruction_id': 'num_Instruction_ids',
                            'id': 'num_posts'}) \
           .sort_values('num_Instruction_ids', ascending=False)

# show top 5 records
cat_df.head(5)
```

Out[14]:

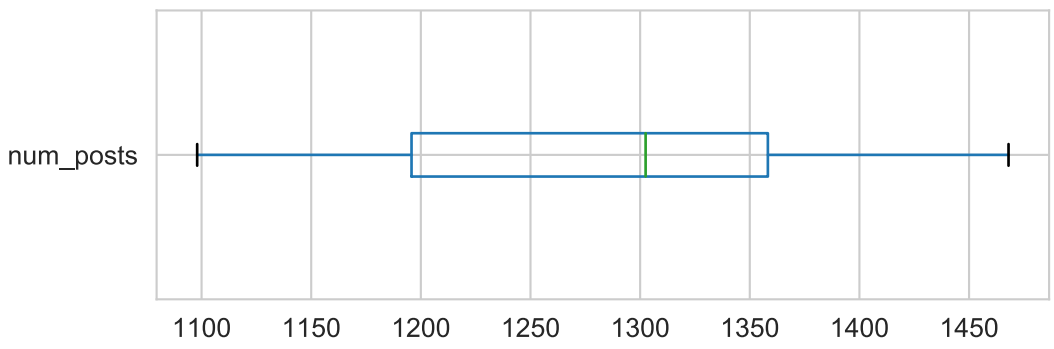
service_type	num_Instruction_ids	num_posts
movie	4	1305
restaurant	3	1300
coffee	2	1376
pizza	2	1468
uber	2	1098

In [15]:
```
cat_df.describe()
```

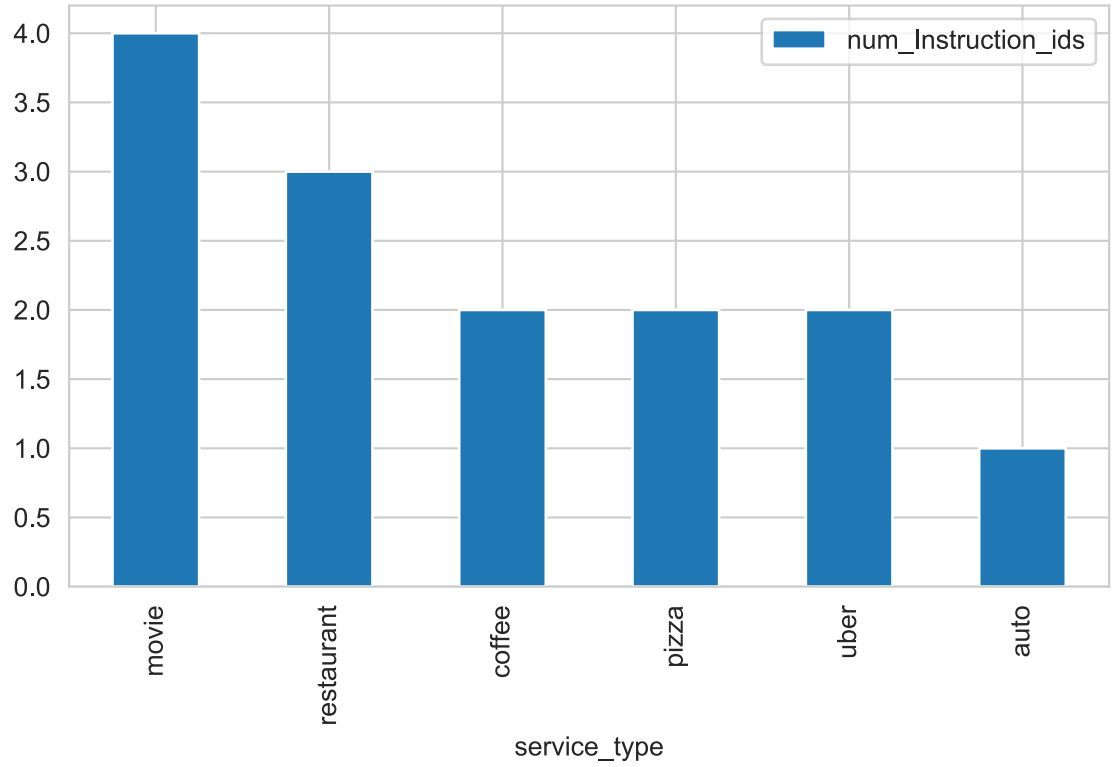
Out[15]:

	num_Instruction_ids	num_posts
count	6.00	6.00
mean	2.33	1284.67
std	1.03	136.19
min	1.00	1098.00
25%	2.00	1195.75
50%	2.00	1302.50
75%	2.75	1358.25
max	4.00	1468.00

In [16]:
```
# horizontal boxplot of a dataframe column
cat_df[['num_posts']].plot(kind='box', vert=False, figsize=(6, 2));
```



In [17]:
```
# bar chart of a dataframe column
cat_df[['num_Instruction_ids']].plot(kind='bar', figsize=(7,4));
```



Word Exploration

In [18]:
```
# create a data frame slice
sub_df = df[df['Instruction_id']=='movie-finder']

# sample cleaned text and tokens tagged as nouns
sub_df[['selfdialog_clean', 'selfdialog_nouns']].sample(2)
```

Out[18]:

	selfdialog_clean	selfdialog_nouns
7003	I feel like watching a documentary What documentary would you like to watch ? Something that has to do with goverment I can pull up a few for you Sure Here are a few i found Now narrow them to the top 5 Got it Which one would you recommend ? I cannot recommend one as they are all equally good Which one has the highest ratings then There are 2 with both the same ratings Ok narrow it down to those I have narrowed it down Ok which has the shortest play time That would be the 9-11 documentary Sounds good Thank you for calling No thank you Goodbye	documentary what documentary something goverment sure top rating rating play time documentary goodbye
880	Hi! I want to find a movie to watch. Hi! What kind of movie do you like to watch? Fantasy movie. Please, name one. What about Hobbit? These are movies about Hobbit. I'd like to watch Unexpected Journey. Do you have something else? These are movies close to your request. Well, I changed my mind. I think about Star Wars now. This is a list of Star Wars movies. Do you want to see any movie now? Can you find something like Star Wars? These are movies close to Star Wars. I would like to see reviews for Return of Jedi. These are Return of Jedi reviews. I would like to see Return of Jedi. Do you want to watch it now? Yes. Do you want to add more movie? No, that's it for now. Enjoy your movie. Good night.	movie kind movie movie what hobbit movie hobbit journey something movie request mind star wars list star wars movie movie something star wars movie star wars review return jedi return jedi return jedi movie movie night

Creating Token List

In [19]:
```
def my_tokenizer(text):
    return text.split() if text != None else []
```

In [20]:
```
# transform list of documents into a single list of tokens
tokens = sub_df.selfdialog_nouns.map(my_tokenizer).sum()
```

In [21]:
```
from collections import Counter

counter = Counter(tokens)
counter.most_common(20)
```

Out[21]:
```
[('movie', 258),
 ('what', 82),
 ('something', 57),
 ('action', 37),
 ('comedy', 35),
 ('tonight', 30),
 ('one', 26),
 ('mood', 22),
 ('film', 22),
 ('time', 20),
 ('netflix', 20),
 ('genre', 19),
 ('star', 19),
 ('anything', 18),
 ('thank', 18),
 ('suggestion', 16),
 ('kind', 15),
 ('rating', 15),
 ('year', 14),
 ('preference', 14)]
```

In [22]:
```
df.service_type.unique()
```

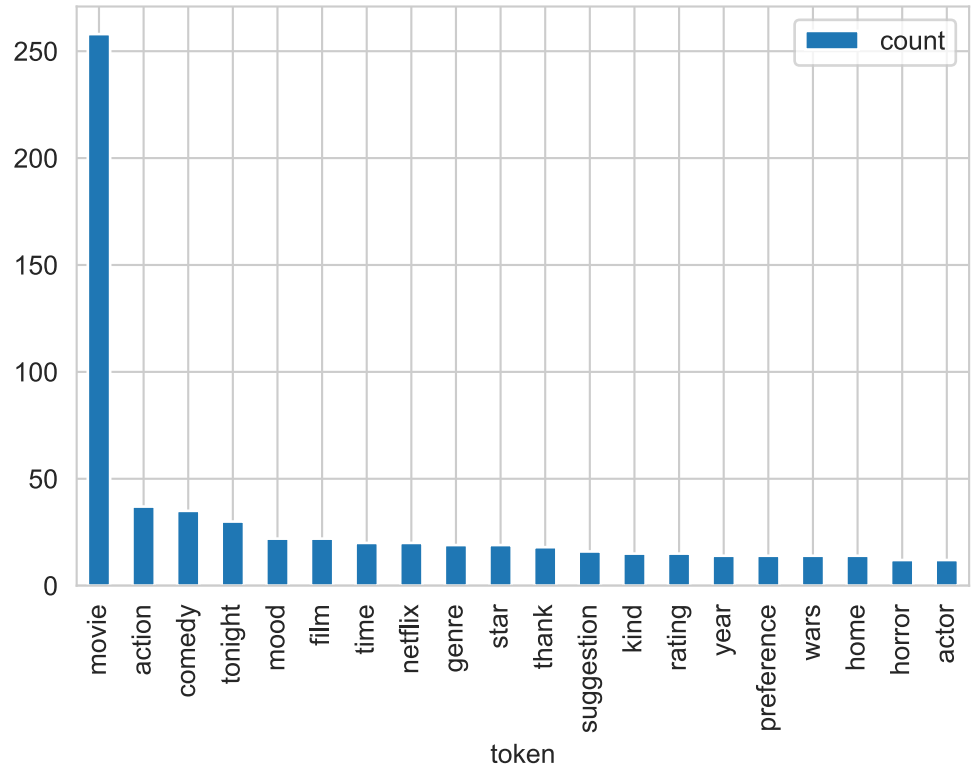
Out[22]:
```
array(['restaurant', 'movie', 'pizza', 'coffee', 'auto', 'uber'],
      dtype=object)
```

In [23]:
```
print([t[0] for t in counter.most_common(200)])
```

```
['movie', 'what', 'something', 'action', 'comedy', 'tonight', 'one', 'mood', 'f
ilm', 'time', 'netflix', 'genre', 'star', 'anything', 'thank', 'suggestion', 'k
ind', 'rating', 'year', 'preference', 'wars', 'home', 'horror', 'actor', 'hou
r', 'ticket', 'assistant', 'problem', 'sci', 'fi', 'sir', 'list', 'who', 'theat
er', 'recommendation', 'day', 'lot', 'scifi', 'world', 'type', 'drama', 'imdb',
'fan', 'mind', 'tom', 'y', 'jedi', 'night', 'john', 'black', 'minute', 'traile
r', 'episode', 'alien', 'romance', 'thriller', 'documentary', 'choice', 'plac
e', 'help', 'nothing', 'book', 'amazon', 'name', 'return', 'wick', 'panther',
'popcorn', 'rush', 'director', 'blade', 'runner', 'release', 'master', 'jurassi
c', 'hanks', 'show', 'marvel', 'mission', 'classic', 'quiet', 'cast', 'box', 'r
eview', 'showing', 'text', 'adam', 'option', 'way', 'adventure', 'empire', 'gal
axy', 'war', 'rosemary', 'baby', 'demand', 'kung', 'fu', 'man', 'avengers',
'x', 'men', 'weather', 'fantasy', 'theme', 'space', 'hero', 'crime', 'wife', 't
oday', 'christmas', 'airplane', 'month', 'incredibles', 'thing', 'table', 'read
y', 'player', 'people', 'matrix', 'bit', 'superhero', 'sandler', 'deadpool', 'p
lotline', 'alright', 'question', 'service', 'mystery', 'science', 'other', 'tho
r', 'art', 'jackie', 'chan', 'yoga', 'kingdom', 'great', 'title', 'infinity',
'link', 'watch', 'chris', 'fiction', 'got', 'mail', 'ryan', 'travel', 'angry',
'story', 'wave', 'idea', 'kong', 'island', 'crazy', 'rich', 'asians', 'jon', 'b
ernthal', 'matter', 'sequel', 'category', 'family', 'earth', 'big', 'violence',
'mrs.', 'stadium', 'style', 'seating', 'guy', 'body', 'part', '1990', 'true',
'lies', 'hobbit', 'laura', 'sure', 'second', "o'clock", 'half', 'actress', 'set
h', 'rogan', 'acting', 'lead', 'buddy', 'cop', 'drug', 'keanu', 'ronin', 'iv',
'new', 'hope', 'harrison', 'ford', 'back', 'james', 'cameron']
```

In [24]:
```
from spacy.lang.en.stop_words import STOP_WORDS

def remove_stopwords(tokens):
    """Remove stopwords from a list of tokens."""
    return [t for t in tokens if t not in STOP_WORDS]

# rebuild counter
counter = Counter(remove_stopwords(tokens))
```

In [25]:
```
# convert list of tuples into data frame
freq_df = pd.DataFrame.from_records(counter.most_common(20),
                                    columns=['token', 'count'])

# create bar plot
freq_df.plot(kind='bar', x='token');
```

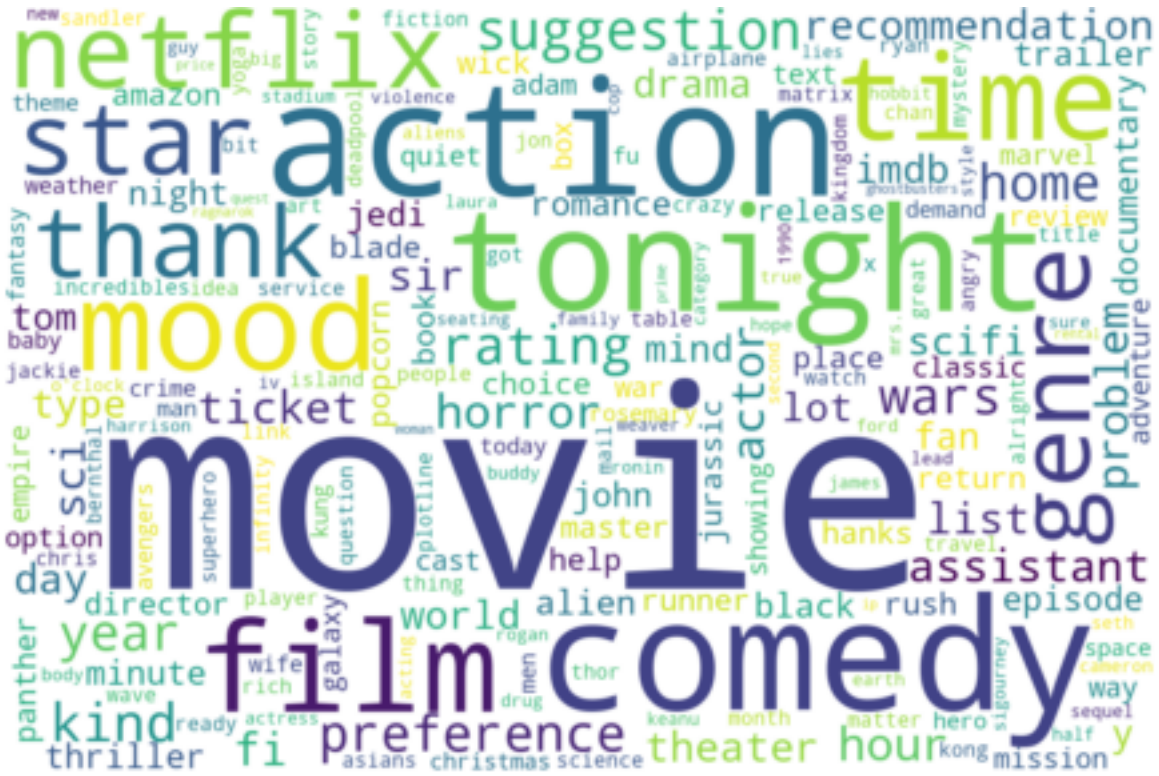


Exploring Word Clouds

In [26]:
```
%matplotlib inline
import matplotlib.pyplot as plt
```

In [27]:
```
from wordcloud import WordCloud

def wordcloud(counter):
    """A small wordloud wrapper"""
    wc = WordCloud(width=1200, height=800,
                   background_color="white",
                   max_words=200)
    wc.generate_from_frequencies(counter)

    # Plot
    fig=plt.figure(figsize=(6, 4))
    plt.imshow(wc, interpolation='bilinear')
    plt.axis("off")
    plt.tight_layout(pad=0)
    plt.show()
```

In [28]:
```
# create wordcloud
wordcloud(counter)
```

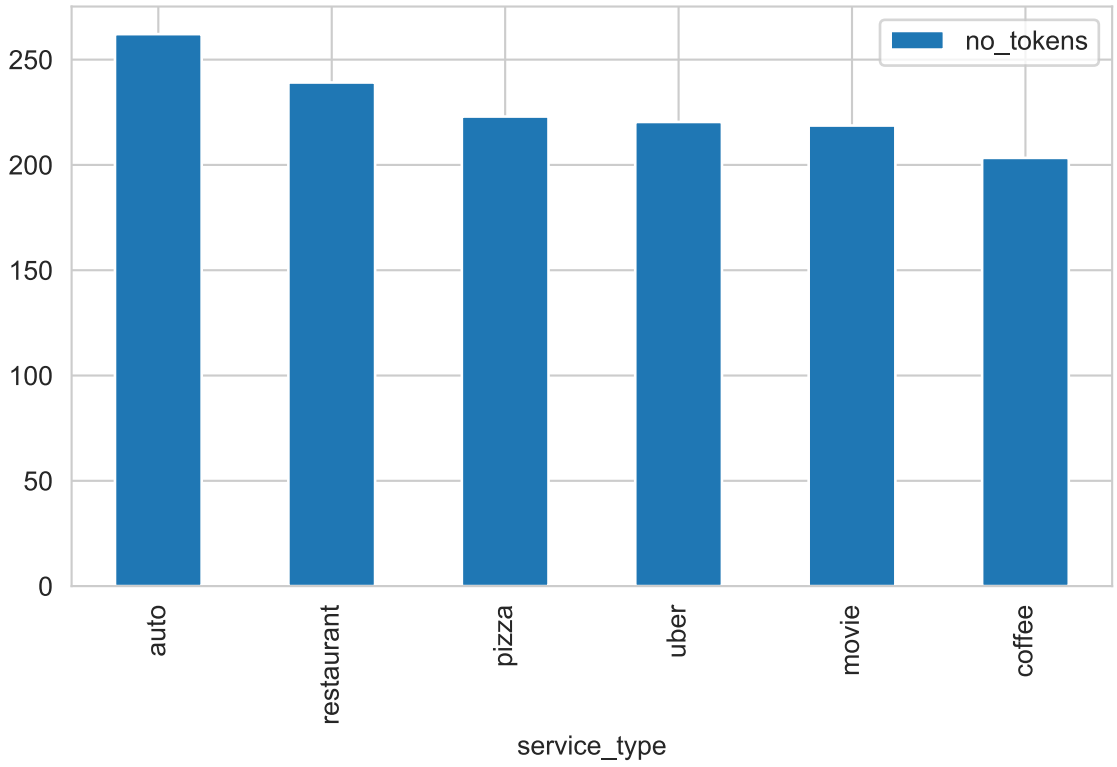


In [29]:
```
tokens2 = df[df['Instruction_id']=='restaurant-table-3'].selfdialog_nouns \
          .map(my_tokenizer).sum()

counter2 = Counter(remove_stopwords(tokens2))
wordcloud(counter2)
```

Exploring Complexity

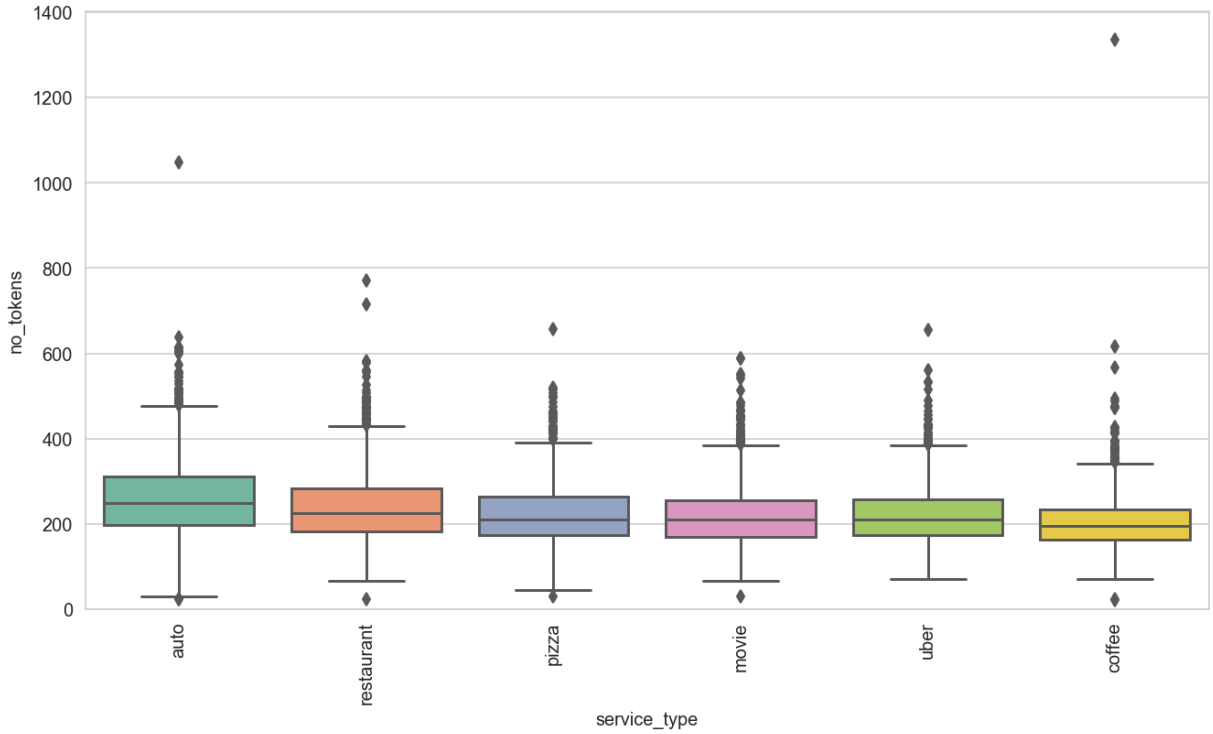
In [30]:
```
df['no_tokens'] = df.selfdialog_lemma\
  .map(lambda l: 0 if l==None else len(l.split()))
```

In [31]:
```
# mean number of tokens by category
df.groupby(['service_type']) \
  .agg({'no_tokens':'mean'}) \
  .sort_values(by='no_tokens', ascending=False) \
  .plot(kind='bar', figsize=(7,4));
```



In [32]:
```
# render plots as retina or png, because svg is very slow
%config InlineBackend.figure_format = 'retina'

import seaborn as sns

def multi_boxplot(data, x, y, ylim = None):
    '''Wrapper for sns boxplot with cut-off functionality'''
    # plt.figure(figsize=(30, 5))
    fig, ax = plt.subplots()
    plt.xticks(rotation=90)

    # order boxplots by median
    ordered_values = data.groupby(x)[[y]] \
                         .median() \
                         .sort_values(y, ascending=False) \
                         .index

    sns.boxplot(x=x, y=y, data=data, palette='Set2',
                order=ordered_values)

    fig.set_size_inches(11, 6)

    # cut-off y-axis at value ylim
    ax.set_ylim(0, ylim)
```

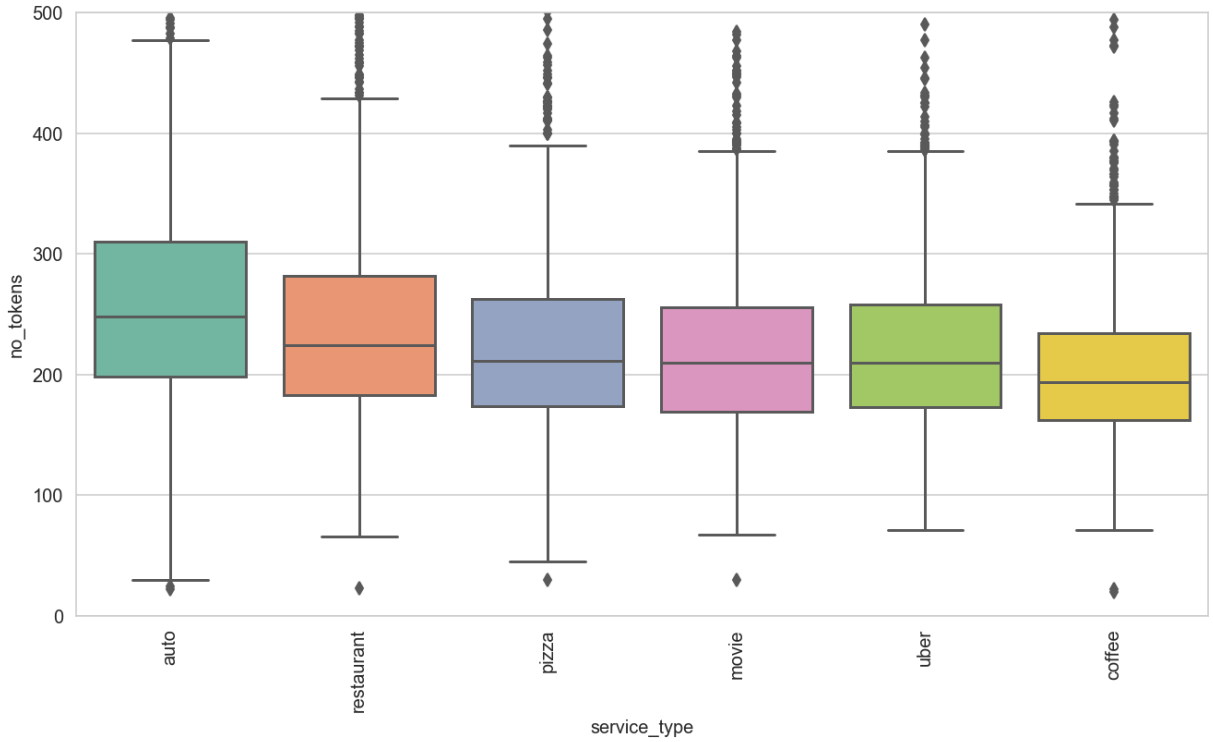
In [33]:
```
multi_boxplot(df, 'service_type', 'no_tokens');
```



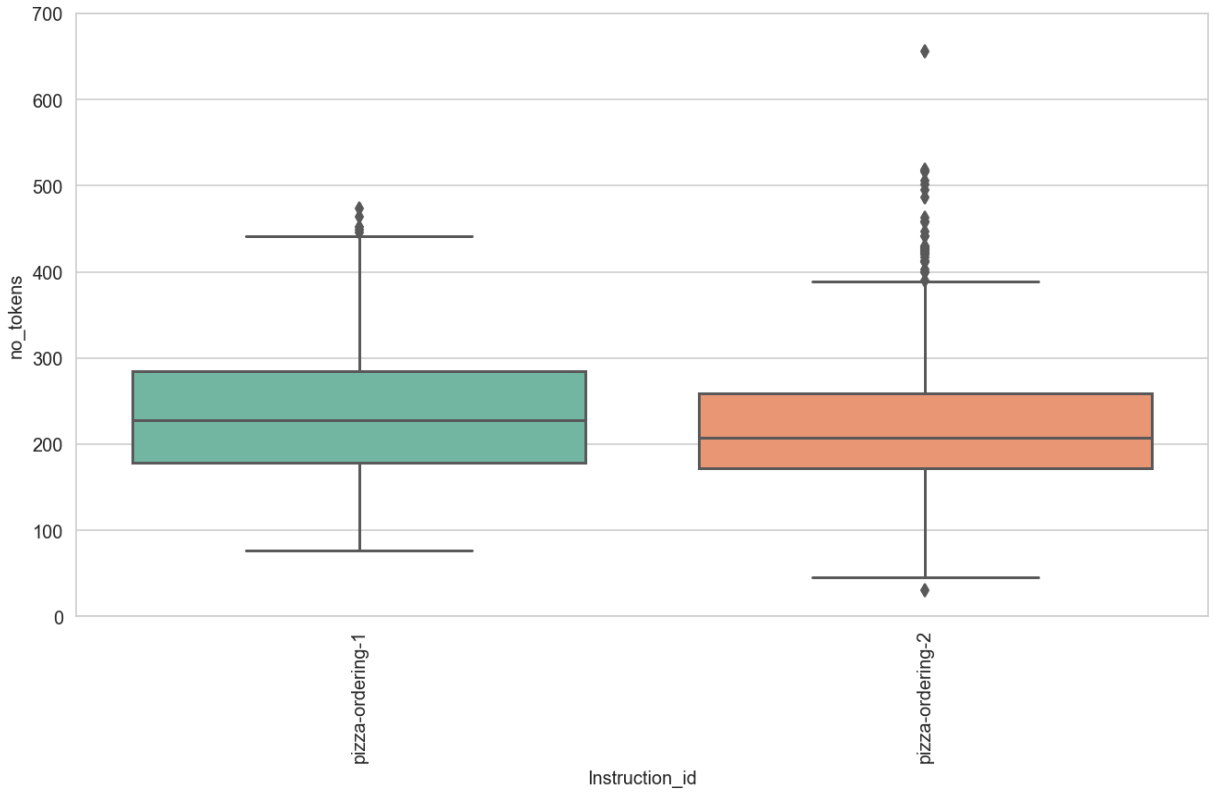
In [34]:
```
# print text of outliers
df['selfdialog_lemma'][df.no_tokens > 1500]
```

Out[34]:
```
Series([], Name: selfdialog_lemma, dtype: object)
```

In [35]:
```
# cut-off diagram at y=500
multi_boxplot(df, 'service_type', 'no_tokens', ylim=500)
```



In [36]:
```
# comparing Instruction_id within a single service_type
multi_boxplot(df[df.service_type=='pizza'],
              'Instruction_id', 'no_tokens', ylim=700)
```



In [ ]: