PROCEDURE OVERVIEW

1. Project directory

G:\\SmartHome

2. Data source

TABLE: reddit_comments

id (PK)	link_id (FK)	parent_id	created_utc	body	author	permalink	score	subreddit
String: comment identifier	String: submission identifier	String: parent of the comment	Unix epoch time: date and time of the comment	String: comment	String: username of the author	String: html link to the comment	Integer: votes given to the comment	String: smarthome or homeautomation

id (PK)	link_id(FK)	created_utc	title	selftext	author	permalink	score	subreddit	num_comments
String: submission identifier (id)	String: add 't3_' to 'id' to make link id	Unix epoch time: date and time of the submission	String: submission title	String: submission self text	String: username of the author	String: html link to the submission	Integer: votes given to the submission	String: smarthome or homeautomation	Integer: number of comments received until data extraction

TABLE: reddit_submissions

DATABASE: reddit_smarthome

Inputs:

From reddit_smarthome database:
- reddit_comments
- reddit_submissions

MySQL_data.p

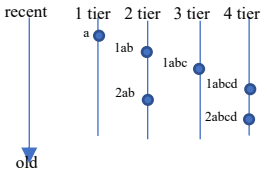
→ Source_Sub_OneTree.py →

- random select 5000 comments From each of the subreddits (smarthome/homeautomation)
- get the corresponding submission and all the comments within the selected comment's tree

Output:

.\\DataSource_backup\\df_tree.csv

Text on the form:
- a tree of comments sorted by tier position and posting time



Output Text: a+1ab+2ab+1abc+1abcd+2abcd

3. Preprocessing

Columns Name	Description – df_tree.csv
tree_ids	comment identifiers separated by <NEW TIER> when the id that follows is a comment from a new tier; or <SAME TIER> when the id that follows is a comment within the last <NEW TIER>
tree_bodies	comments separated by <NEW TIER> when the comment that follows is a comment from a new tier; or <SAME TIER> when the comment that follows is a comment within the last <NEW TIER>
id	first tier comment identifier
link_id	submission identifier
title	submission title
selftext	submission self text

Inputs:

.\\DataSource_backup\\df_tree.csv

NLP_preprocessing.py

Output:

.\\DataSource_backup\\sub_onetree_train.csv

.\\DataSource_backup\\sub_onetree_test.csv

bot_test.py

- remove comments from bots manually identified using bot_test.py

- remove comments where 70% words are not in English

spelling_test.py

- run a spelling check to see If there are systematic errors

clean_text.py

- deal with stop words, URLs, html formatting, Internal hyphen, punctuation, lemmatization, stemming
- remove row with short text
- divide the data in 80% training and 20% testing

4. Visualization

NLP_visualization.py

- words frequency from text
- vocabulary descriptive stats
- word count distribution
- words frequency from vocabulary

5. Modelling

Columns Name	Description – sub_onetree_train.csv
tree_ids	comment identifiers separated by <NEW TIER> when the id that follows is a comment from a new tier; or <SAME TIER> when the id that follows is a comment within the last <NEW TIER>
tree_bodies	comments separated by <NEW TIER> when the comment that follows is a comment from a new tier; or <SAME TIER> when the comment that follows is a comment within the last <NEW TIER>
id	first tier comment identifier
link_id	submission identifier
title	submission title
selftext	submission self text
text	submission title <SUB> submission selftext <SUB> tree_bodies
URL	stripped out hyperlinks
clean_text	columnn with the pre-processed text

Inputs:
.\\DataSource_backup
sub_onetree_train.csv

→ NLP_modelling.py →

Output:
- trained vocabulary: **nb**xx_**na**xx
  [**nb**: no_below, **na**: no_above]
- trained models dict : **a**xx_**b**xx
  [**a**: alpha, **b**: beta]
- trained bigram

- create bi-gram
- save trained bigram in \\venv\\lib\\site-packages\\ gensim\\test\\test_data\\train_bigram\\ nbxx_naxx_bigram.pkl
- remove from the vocabulary words that occur too often and too infrequently
- save vocabulary in \\venv\\lib\\site-packages\\ gensim\\test\\test_data\\vocabulary\\nbxx_naxx
- run models and save in \\venv\\lib\\site-packages\\ gensim\\test\\test_data\\train_models\\ nbxx_naxx_axxx_bxxx_models.pkl

NOTE: working with 8GB RAM

6. Evaluation

Inputs:
.\\DataSource_backup
sub_onetree_train.csv
trained vocabulary
trained models
trained bigram

→ NLP_evaluation.py →

Output:
- In \\venv\\lib\\site-packages\\gensim\\ test\\test_data\\evaluation\\
- trained models dict : **a**xx_**b**xx
  [**a**: alpha, **b**: beta]

- Calculate Coherence Gensim cv
- Calculate Cao Juan 2009
- Calculate Arun 2010
- Calculate Coherence Mimno 2011

IMPORTANT: run evaluation from terminal with args: the trained vocabulary of interest and the alpha params
> python NLP_evaluation.py nb5_na04 a001_
> python NLP_evaluation.py nb5_na04 a01_
> python NLP_evaluation.py nb5_na04 a1_
> python NLP_evaluation.py nb5_na04 a10_

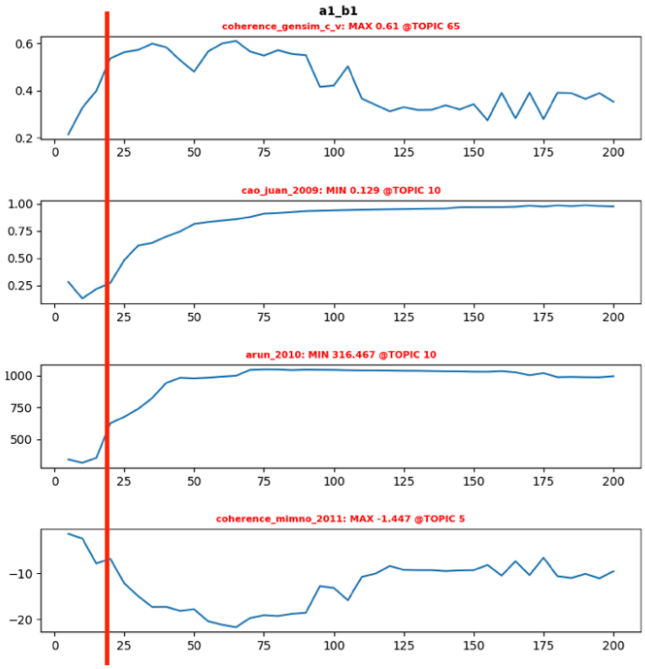
7. Selection

Inputs:
evaluation metrics in
\\venv\\lib\\site-packages\\gensim\\
test\\test_data\\evaluation\\

→ NLP_selection.py →

Output:
- In .*Figure*\\

- plot the evaluation metrics trends in order to find the best combination of alpha, beta and num. of topics



7. Model Inspection

Inputs:
selected trained model
trained vocabulary
trained bigram
.\\DataSource_backup\\sub_onetree_train.csv

→ LDA_classification.py →

NLP_inspection.ipynb/
NLP_inspection.html

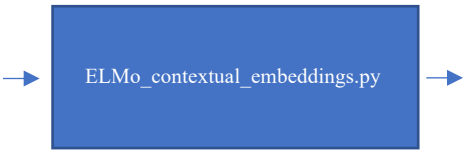
Output:
In \\venv\\lib\\site-packages\\gensim\\
test\\test_data\\inspection\\
- JSD calculation for all documents in nb5_na04_JSD_dict.pkl
- Documents with JSD <= 0.4 to the to the reference for each topic

- top words per topic
- pyLDAvis
- summary output documents/topics
- Jensen-Shannon Distance

Index pyLDAvis	Inferred topic	LDA topic number
1	Broad topic regarding Automation - Devices - Network	19
2	Smart lights	18
3	Smart termostat	9
4	Home entertainment - voice assistant	7
5	Audio - Speakers	4
6	Smart Lock systems	16
8	Smart camera - surveillance	15
9	Smart plugs - power systems	11
10	Smart door systems	5

8. Semantic Search Engine

Inputs:
Documents with JSD <= 0.4 to the
to the reference for each topic



Output:

- Semantic Search Engine
- Cosine Similarity