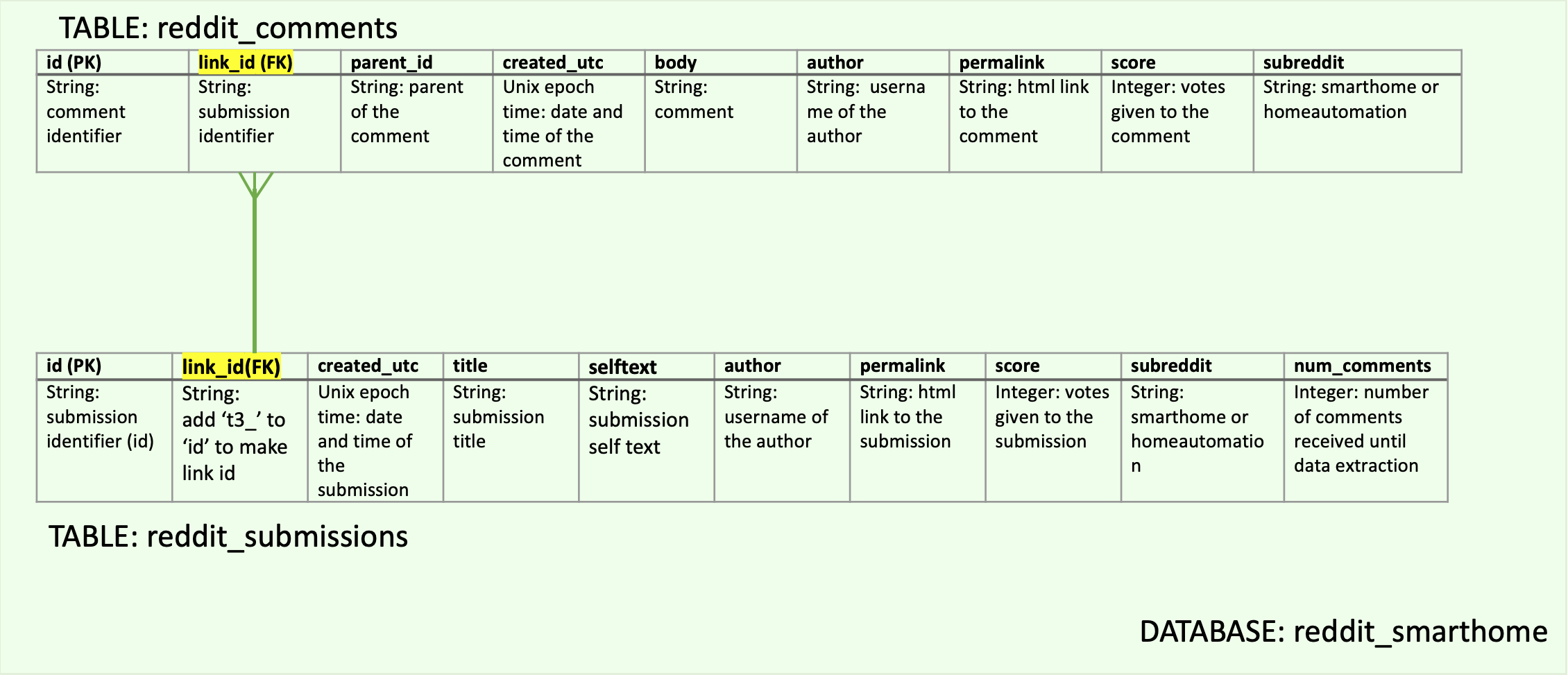
**PROCEDURE OVERVIEW**

**1. Project directory**

G:\\SmartHome

**2. Data source**



**Inputs: Output:**

From reddit\_smarthome database: *.\\DataSource\_backup\\df\_tree.csv*

Source\_Sub\_OneTree.py

- reddit\_comments Text on the form:

- reddit\_submissions - a tree of comments sorted by tier

position and posting time

MySQL\_data.py

recent

4 tier

3 tier

2 tier

1 tier

- random select 5000 comments

1ab

a

From each of the subreddits

1abc

(smarthome/homeautomation)

1abcd

2ab

- get the corresponding submission

2abcd

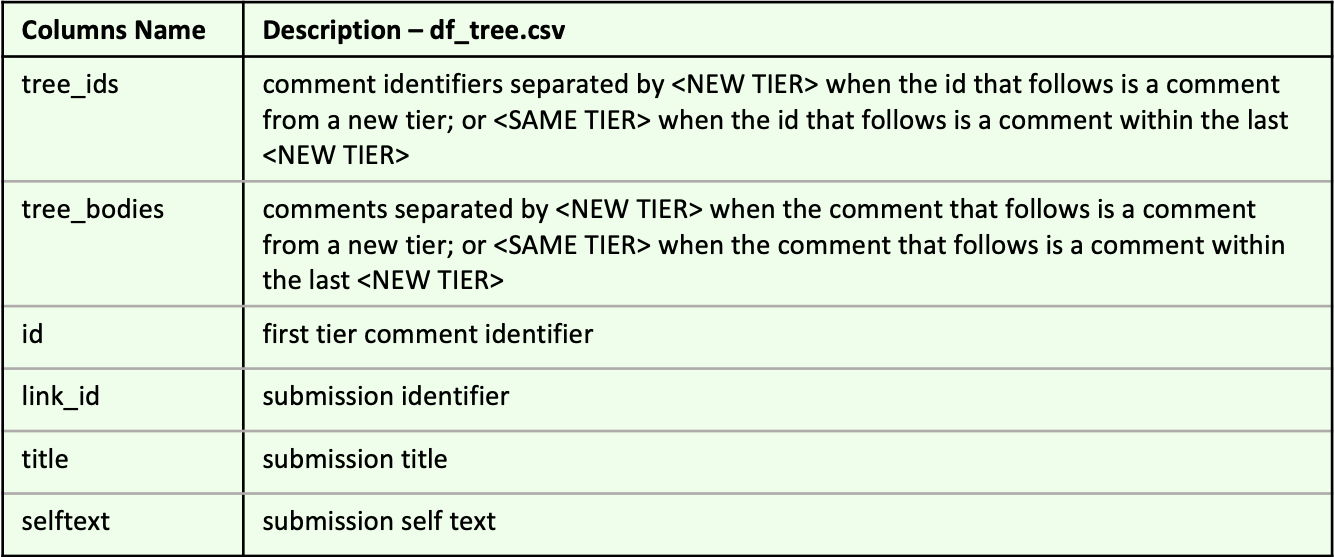
and all the comments within the

old

selected comment’s tree

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

**3. Preprocessing**

****

**Inputs: Output:**

NLP\_preprocessing.py

*.\\DataSource\_backup\\df\_tree.csv* *.\\DataSource\_backup\\*

bot\_test.py

*sub\_onetree\_train.csv*

*.\\DataSource\_backup\\*

*sub\_onetree\_test.csv*

- remove comments from bots

manually identified using bot\_test.py

- remove comments where 70% words

are not in English

spelling\_test.py

clean\_text.py

- run a spelling check to see

If there are systematic errors

- 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

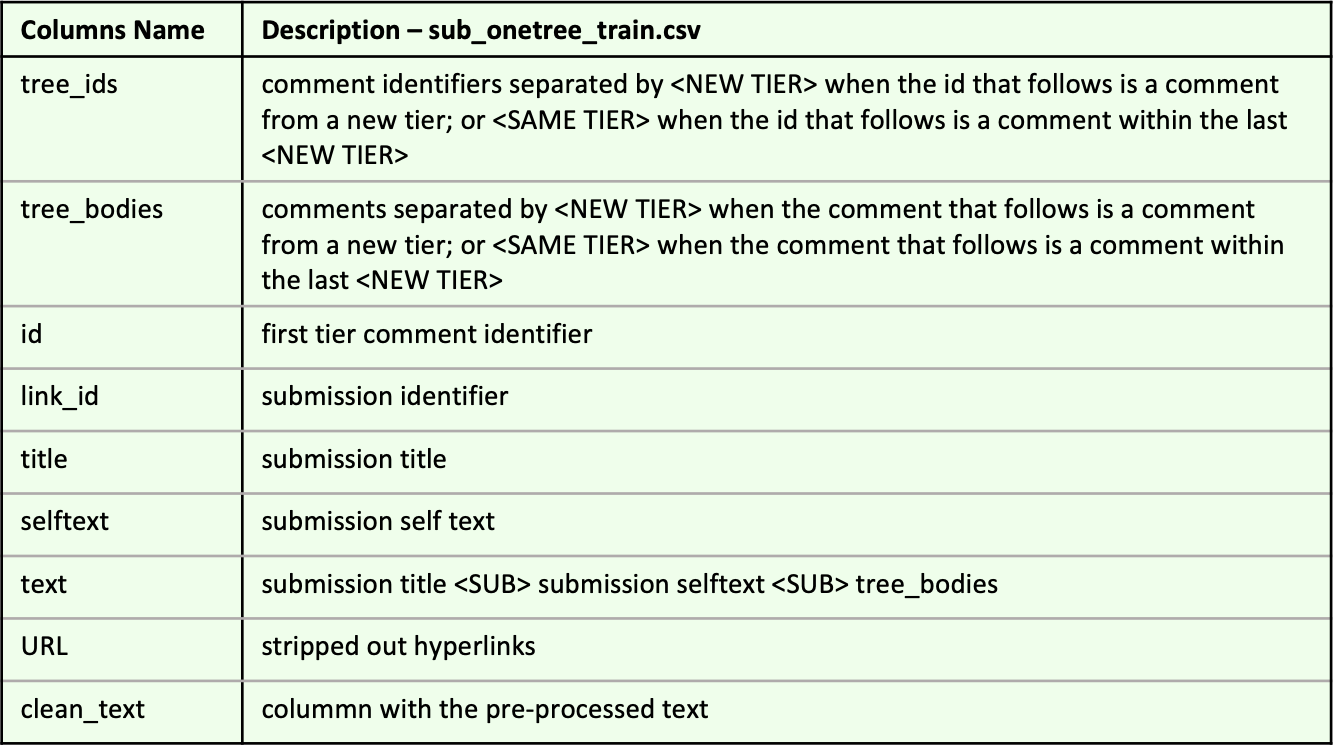
**4. Visualization**

NLP\_visualization.py

**-** words frequency from text - word count distribution

- vocabulary descriptive stats - words frequency from vocabulary

**5. Modelling**

****

NLP\_modelling.py

**Inputs: Output:**

*.\\DataSource\_backup\\* - trained vocabulary: **nb**xx\_**na**xx

*sub\_onetree\_train.csv* [**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\\](file://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\\](file://venv//lib//site-packages//)

gensim\\test\\test\_data\\vocabulary\\nbxx\_naxx

- run models and save in [\\venv\\lib\\site-packages\\](file://venv//lib//site-packages//)

gensim\\test\\test\_data\\train\_models\\ nbxx\_naxx\_axxx\_bxxx\_models.pkl

NOTE: working with 8GB RAM

**6. Evaluation**

NLP\_evaluation.py

**Inputs: Output:**

*.\\DataSource\_backup\\* - In [\\venv\\lib\\site-packages\\](file://venv//lib//site-packages//)gensim\\

*sub\_onetree\_train.csv* test\\test\_data\\evaluation\\

trained vocabulary - trained models dict : **a**xx\_**b**xx

trained models - Calculate Coherence Gensim cv [**a**: alpha, **b**: beta]

trained bigram - 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**

NLP\_selection.py

**Inputs: Output:**

evaluation metrics in - In *.\\Figure\\*

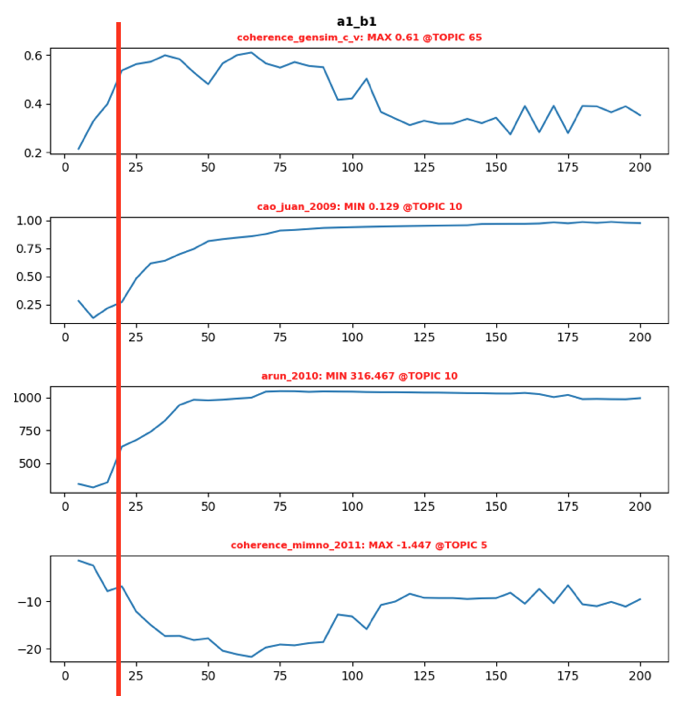
[\\venv\\lib\\site-packages\\](file://venv//lib//site-packages//)gensim\\

test\\test\_data\\evaluation\\

- plot the evaluation metrics trends

in order to find the best combination

of alpha, beta and num. of topics



**7. Model Inspection**

LDA\_classification.py

**Inputs: Output:**

selected trained model In [\\venv\\lib\\site-packages\\](file://venv//lib//site-packages//)gensim\\

trained vocabulary test\\test\_data\\inspection\\

trained bigram - JSD calculation for all documents in

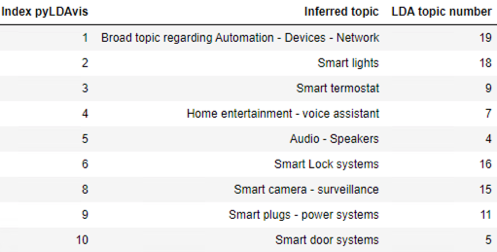
*.\\DataSource\_backup\\* nb5\_na04\_JSD\_dict.pkl

NLP\_inspection.ipynb/

NLP\_inspection.html

*sub\_onetree\_train.csv* - 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

**8. Semantic Search Engine**

**Inputs: Output:**

ELMo\_contextual\_embeddings.py

Documents with JSD <= 0.4 to the

to the reference for each topic

- Semantic Search Engine

- Cosine Similarity