[(0, '0.010\*"азербайджан" + 0.009\*"тюркский" + 0.008\*"тюрчанки" + 0.007\*"женский" + 0.007\*"мужчина" + 0.006\*"клуб" + 0.006\*"масса" + 0.006\*"тюрчанок" + 0.005\*"девочка" + 0.004\*"положение"'),

(1, '0.013\*"чувашка" + 0.012\*"чувашский" + 0.008\*"чуваш" + 0.004\*"революция" + 0.004\*"деревня" + 0.003\*"население" + 0.003\*"количество" + 0.003\*"республика" + 0.003\*"обычай" + 0.003\*"больший"'),

(2, '0.007\*"население" + 0.005\*"женский" + 0.004\*"ребёнок" + 0.004\*"являться" + 0.004\*"татарский" + 0.004\*"край" + 0.004\*"муж" + 0.004\*"область" + 0.004\*"образ" + 0.004\*"власть"'),

(3, '0.000\*"китайский" + 0.000\*"китай" + 0.000\*"женский" + 0.000\*"являться" + 0.000\*"население" + 0.000\*"ребёнок" + 0.000\*"область" + 0.000\*"мужчина" + 0.000\*"муж" + 0.000\*"девушка"'),

(4, '0.016\*"дагестан" + 0.014\*"горянка" + 0.006\*"горец" + 0.005\*"невеста" + 0.005\*"дагестанский" + 0.004\*"родитель" + 0.004\*"аул" + 0.003\*"муж" + 0.003\*"женотдел" + 0.003\*"язык"'),

(5, '0.008\*"чеченский" + 0.007\*"чеченец" + 0.007\*"турецкий" + 0.006\*"турция" + 0.006\*"девушка" + 0.006\*"чеченка" + 0.005\*"являться" + 0.005\*"положение" + 0.005\*"народ" + 0.005\*"женский"'),

(6, '0.005\*"являться" + 0.005\*"персидский" + 0.004\*"население" + 0.004\*"женский" + 0.004\*"ребёнок" + 0.003\*"мордва" + 0.003\*"персия" + 0.003\*"больший" + 0.003\*"муж" + 0.003\*"положение"'),

(7, '0.010\*"курд" + 0.006\*"марийка" + 0.005\*"ребёнок" + 0.004\*"курдский" + 0.004\*"марь" + 0.004\*"население" + 0.004\*"положение" + 0.003\*"быт" + 0.003\*"семья" + 0.003\*"мариек"'),

(8, '0.010\*"китайский" + 0.009\*"китай" + 0.005\*"женский" + 0.005\*"туркмен" + 0.005\*"туркменский" + 0.004\*"рабочий" + 0.004\*"тсср" + 0.004\*"являться" + 0.004\*"муж" + 0.003\*"положение"'),

(9, '0.000\*"население" + 0.000\*"женский" + 0.000\*"ребёнок" + 0.000\*"являться" + 0.000\*"муж" + 0.000\*"образ" + 0.000\*"положение" + 0.000\*"быт" + 0.000\*"отношение" + 0.000\*"мужчина"')]

#import libraries

import spacy

from spacy import displacy

# Import external libraries: gensim to create models and do some additional preprocessing

import gensim

import gensim.corpora as corpora

from gensim.utils import simple\_preprocess

from gensim.models import CoherenceModel

# Import external libraries: pyLDA for vis

import pyLDAvis

import pyLDAvis.gensim\_models as gensimvis

import os

#set up filepaths, source files

import ru\_core\_news\_lg

nlp = ru\_core\_news\_lg.load()

stopwordsfile = 'ru.txt'

outname = 'topicmodel.csv'

# Open and read text my list of stopwords

stopwords = open(stopwordsfile, encoding='utf-8').read()

stopwords\_list = stopwords.splitlines()

#add the my stopwords to spaCy's list

for stopword in stopwords\_list:

word = nlp.vocab[stopword]

word.is\_stop = True

# Define a function to remove stop words from a sentence

def remove\_stop\_words(sentence):

# Parse the sentence using spaCy

doc = nlp(sentence)

# Use a list comprehension to remove stop words

filtered\_tokens = [token for token in doc if not token.is\_stop]

# Join the filtered tokens back into a sentence

return ' '.join([token.text for token in filtered\_tokens])

# Preprocess texts for gensim

def gen\_words(texts): # create a function called gen\_words that runs on a list

final = [] # Create an empty list to hold tokens

for text in texts:

new = gensim.utils.simple\_preprocess(text, deacc = False)

# If working with languages that employ accents, you can set deacc to False

final.append(new)

return (final)

# Load all texts to a list

# Use OS to get a list of every file in the folder

folder\_path = "lemmatized\_files"

# Make an empty list for the corpus

lemmatized\_texts = []

for filename in os.listdir(folder\_path): # For every file in the specified folder

file\_path = os.path.join(folder\_path, filename)

if os.path.isfile(file\_path): # Check if it's a file

# Print file name for debugging (display)

print(f"Processing file: {filename}")

# Open and read the file into a variable text

text = open(file\_path, encoding='utf-8').read()

filtered\_text = remove\_stop\_words(text)

# Append that text varible to corpus emply list

lemmatized\_texts.append(filtered\_text)

data\_words = gen\_words(lemmatized\_texts) # Pass lemmatized\_texts from previous step through the gen\_words function

# Combine bigrams and trigrams

bigram\_phrases = gensim.models.Phrases(data\_words, min\_count=3, threshold=50)

trigram\_phrases = gensim.models.Phrases(bigram\_phrases[data\_words], threshold=50)

bigram = gensim.models.phrases.Phraser(bigram\_phrases)

trigram = gensim.models.phrases.Phraser(trigram\_phrases)

def make\_bigrams(texts):

return [bigram[doc] for doc in texts]

def make\_trigrams(texts):

return [trigram[bigram[doc]] for doc in texts]

data\_bigrams = make\_bigrams(data\_words)

data\_bigrams\_trigrams = make\_trigrams(data\_bigrams)

# --Uncomment to print list of words showing bigrams and trigrams

# print (data\_bigrams\_trigrams[0])

# Create dictionary of all words in texts

id2word = corpora.Dictionary(data\_bigrams\_trigrams)

# Represent dictionary words as tuples (index, frequency)

corpus = []

for text in data\_bigrams\_trigrams:

new = id2word.doc2bow(text)

corpus.append(new)

# Specify number of topics (clusters of words)

num\_topics = 10 # Experiment with more and fewer numbers of topics, comparing results

# Create LDA model

lda\_model = gensim.models.ldamodel.LdaModel(corpus=corpus,

id2word=id2word,

num\_topics=num\_topics,

random\_state=100,

update\_every=1,

chunksize=100,

# Change chunksize to increase or decrease the length of segments

passes=50,

# Can do more passes but will increase the time it takes the block to run

alpha="auto")

# Print topics

print(lda\_model.show\_topics())

# Output visualization

vis\_data = gensimvis.prepare(lda\_model, corpus, id2word, R=15, mds='mmds')

vis\_data

pyLDAvis.display(vis\_data)

pyLDAvis.save\_html(vis\_data, './topicVis' + str(num\_topics) + '.html')