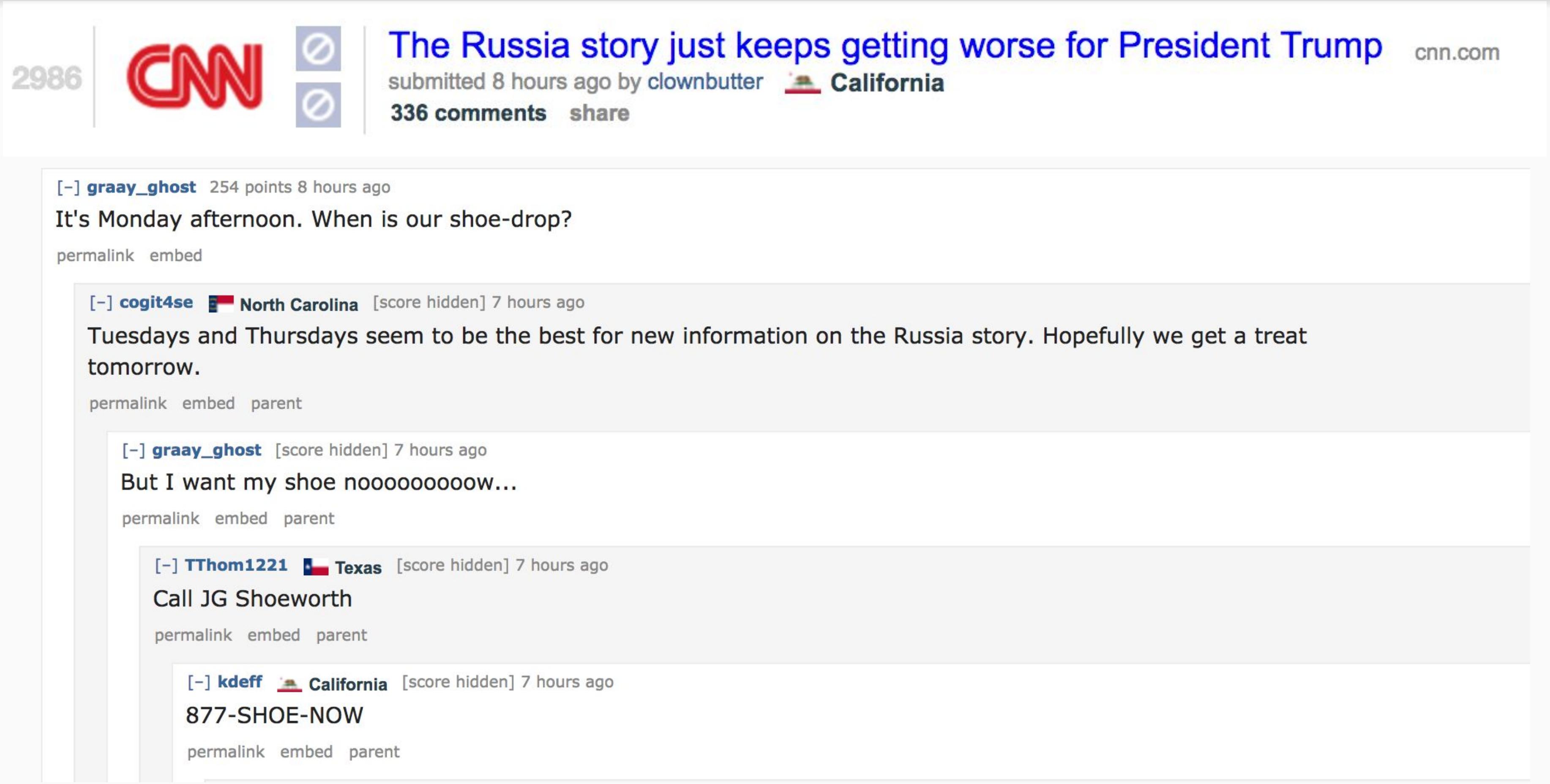
Diversionary comments under political blog posts

Bing Liu
Philip S. Yu

Jing Wang
Weiyi Meng

Diversionary comments are everywhere!



2986

The Russia story just keeps getting worse for President Trump cnn.com

submitted 8 hours ago by clownbutter California

336 comments share

[–] graay_ghost 254 points 8 hours ago

It's Monday afternoon. When is our shoe-drop?

permalink embed

[–] cogit4se North Carolina [score hidden] 7 hours ago

Tuesdays and Thursdays seem to be the best for new information on the Russia story. Hopefully we get a treat tomorrow.

permalink embed parent

[–] graay_ghost [score hidden] 7 hours ago

But I want my shoe noooooooow...

permalink embed parent

[–] TThom1221 Texas [score hidden] 7 hours ago

Call JG Shoeworth

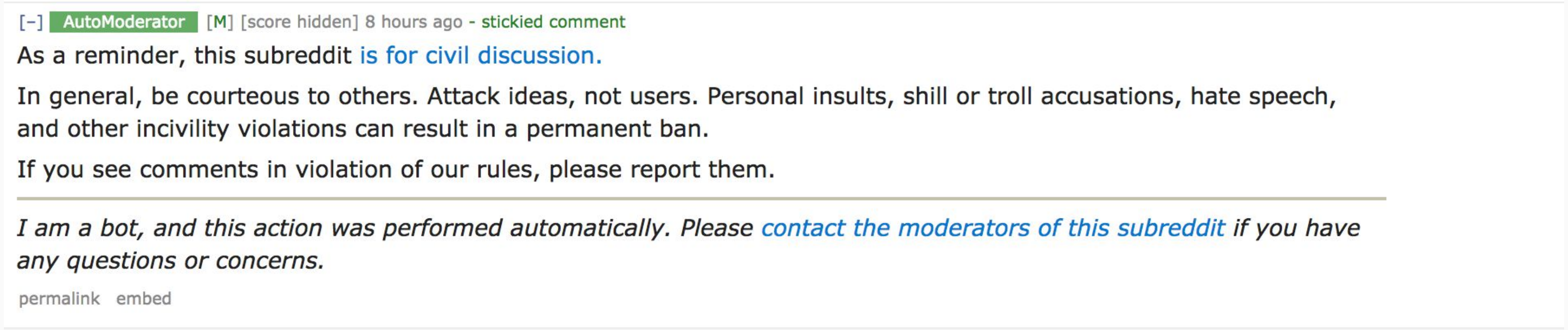
permalink embed parent

[–] kdeff California [score hidden] 7 hours ago

877-SHOE-NOW

permalink embed parent

Diversionary comments are everywhere!



[–] AutoModerator [M] [score hidden] 8 hours ago - stickied comment

As a reminder, this subreddit is for civil discussion.

In general, be courteous to others. Attack ideas, not users. Personal insults, shill or troll accusations, hate speech, and other incivility violations can result in a permanent ban.

If you see comments in violation of our rules, please report them.

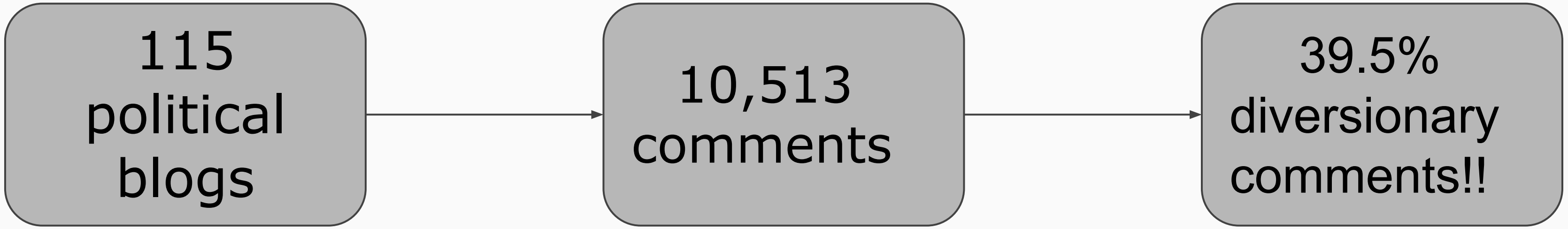
I am a bot, and this action was performed automatically. Please contact the moderators of this subreddit if you have any questions or concerns.

permalink embed

Why study such a problem?

- About 2 million blog posts are written each day!
- People believe that content in the blogosphere is more trustworthy
- These comments deliberately twist the bloggers’ intention.
- Advertise products.
- Confuse the reader about the true nature of the blog

Why study such a problem?



Considerable negative impact!!

They come in all shapes and forms!

1. Shifting to unrelated topics (60%)
2. Personal attacks to commentators (22%)
3. With little content (10%)
4. About the hosting websites (5%)
5. Advertising in comments (3%)

Challenges

- To find an accurate representation for each comment and the post
  - Comments are short and offer little literal information
- Pronouns and hidden knowledge
  - Comments often include political figures or events which are not explicitly mentioned in the post

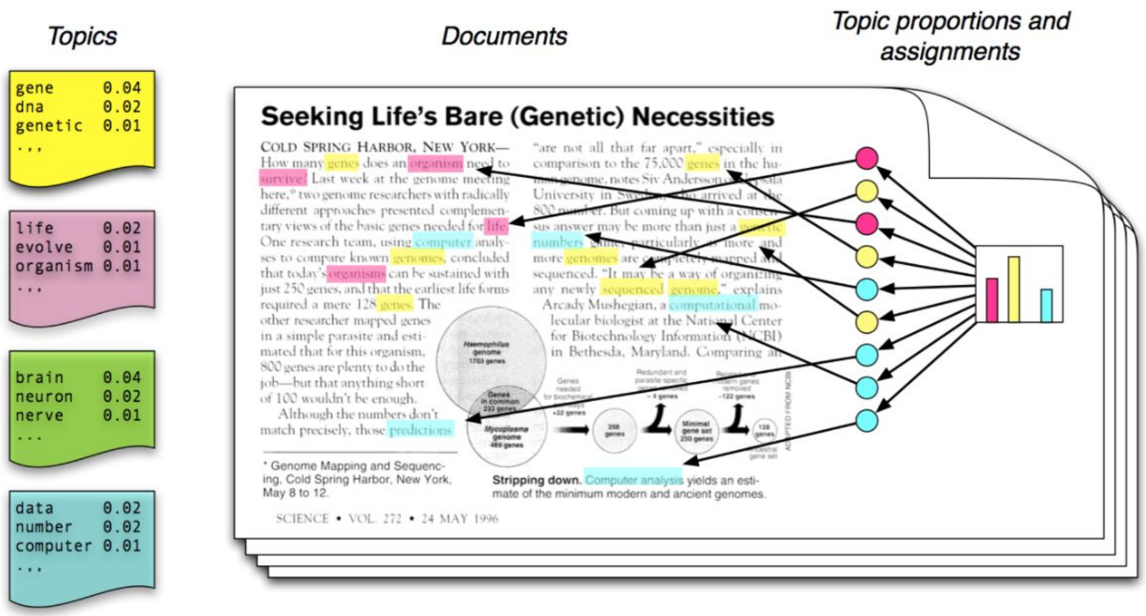
Solutions

Pre-processing steps:

1. Coreference resolution
   a. Obama - He - He - He
2. Extract hidden knowledge from Wikipedia
   a. Pick-up anchor tags from the first paragraph about searched entity
3. Latent Dirichlet Allocation -> topic distributions -> document vectors

Cosine and KL-Divergence to measure relatedness

Latent Dirichlet Allocation



- Each **topic** is a distribution over words
- Each **document** is a mixture of corpus-wide topics
- Each **word** is drawn from one of those topics

Document-topic and term-topic distributions

Document-topic distribution:

$$\Theta = \frac{C_{dj}^{DT} + \alpha}{\sum_{k=1}^{T} C_{dk}^{DT} + T\alpha}$$

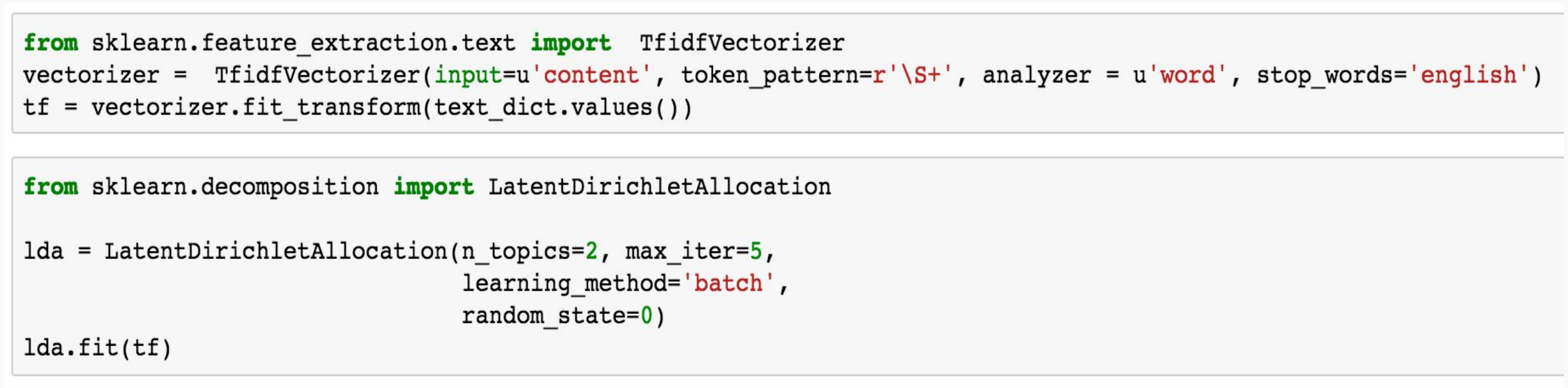
C_{dj}^{DT}	= Total count of words in document d which have been assigned to topic j
α	= Smoothing constant

Term-topic distribution:

$$\varphi = \frac{C_{ij}^{WT} + \beta}{\sum_{k=1}^{W} C_{kj}^{WT} + W\beta}$$

C_{ij}^{WT}	= Number of times word i has been assigned to topic j
β	= Smoothing constant

LDA: Demo example

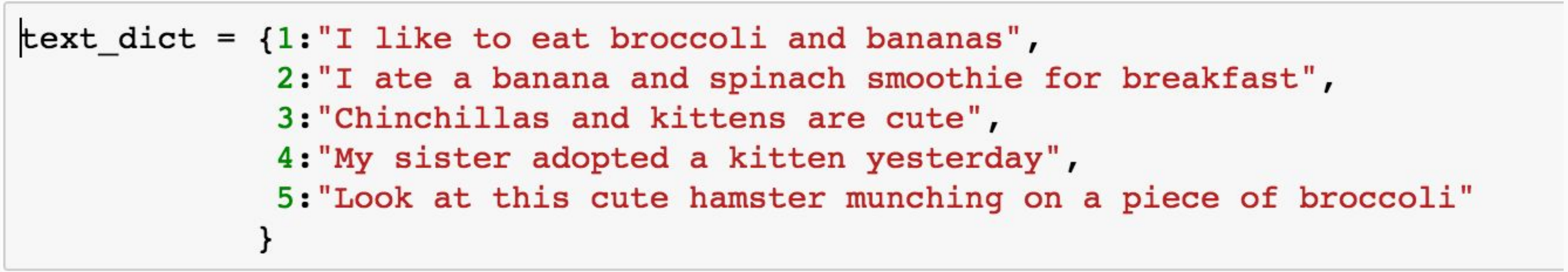


```python
from sklearn.feature_extraction.text import  TfidfVectorizer
vectorizer =  TfidfVectorizer(input=u'content', token_pattern=r'\S+', analyzer = u'word', stop_words='english')
tf = vectorizer.fit_transform(text_dict.values())
```

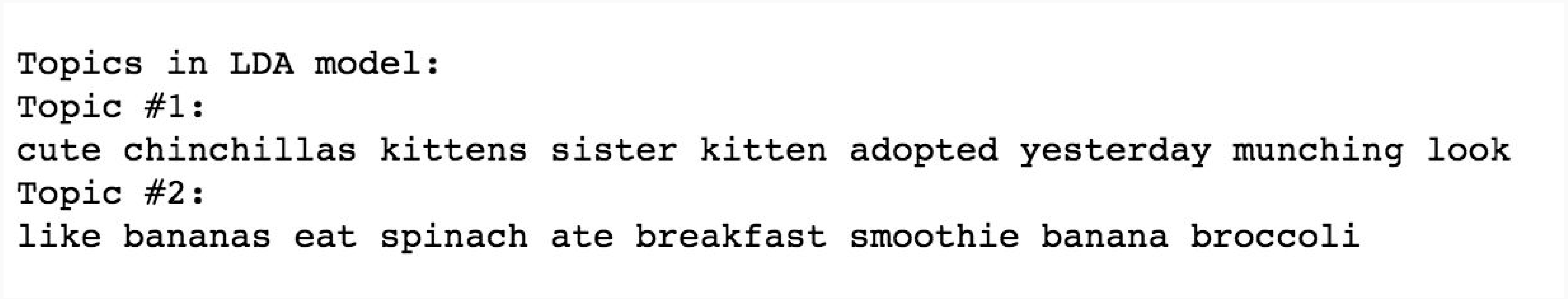
```python
from sklearn.decomposition import LatentDirichletAllocation

lda = LatentDirichletAllocation(n_topics=2, max_iter=5,
                                learning_method='batch',
                                random_state=0)
lda.fit(tf)
```

LDA: Results

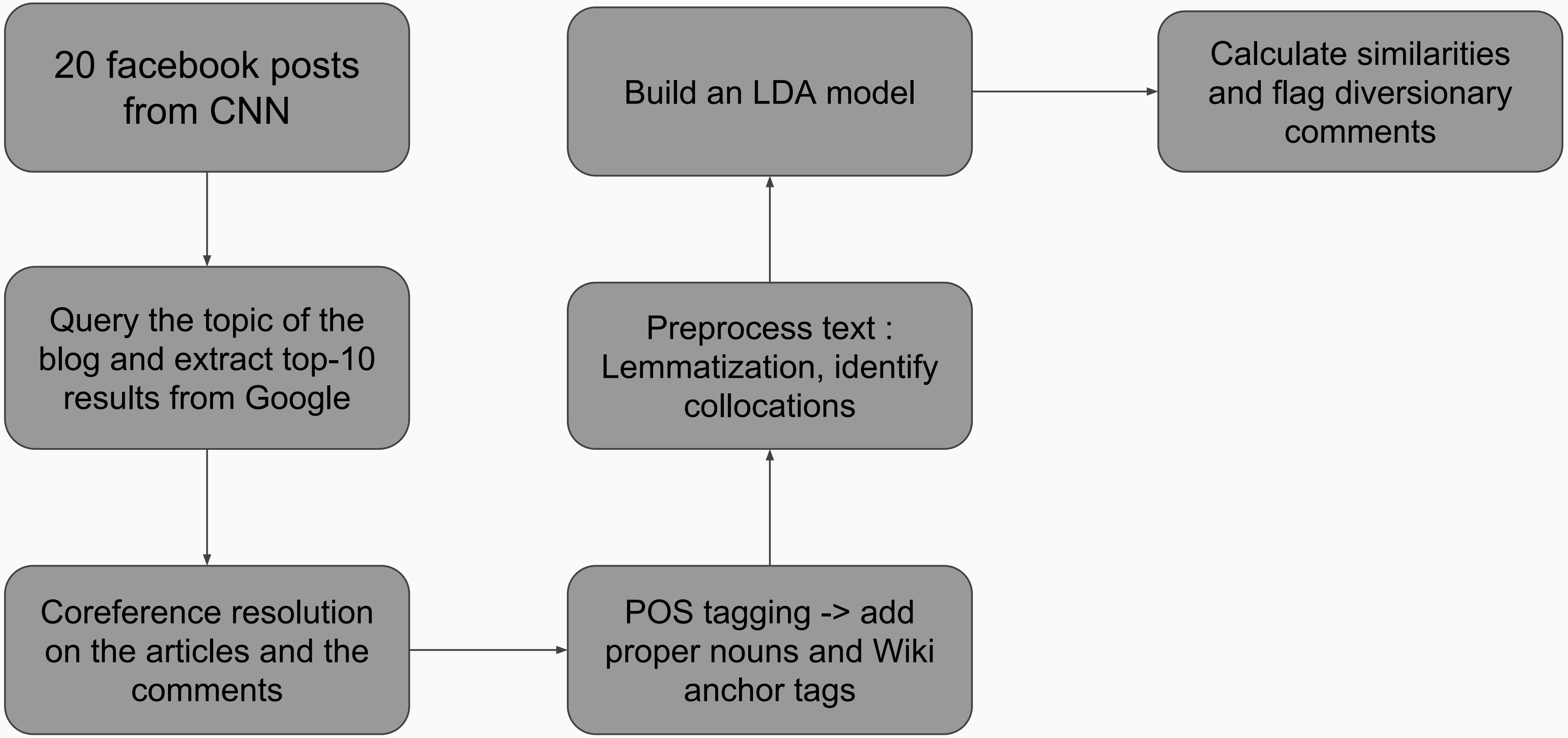


```
text_dict = {1:"I like to eat broccoli and bananas",
             2:"I ate a banana and spinach smoothie for breakfast",
             3:"Chinchillas and kittens are cute",
             4:"My sister adopted a kitten yesterday",
             5:"Look at this cute hamster munching on a piece of broccoli"
            }
```



```
Topics in LDA model:
Topic #1:
cute chinchillas kittens sister kitten adopted yesterday munching look
Topic #2:
like bananas eat spinach ate breakfast smoothie banana broccoli
```

The Pipeline


20 facebook posts from CNN
Query the topic of the blog and extract top-10 results from Google
Coreference resolution on the articles and the comments
POS tagging -> add proper nouns and Wiki anchor tags
Preprocess text : Lemmatization, identify collocations
Build an LDA model
Calculate similarities and flag diversionary comments

Fetching the data

- Created our own dataset from scratch
  - CNN news page on Facebook
- Query the title of each post
  - Google Custom Search Engine to extract the top 10 results
  - Newspaper python module to extract the article from each link

Coreference resolution

- Groups all the mentioned entities in a document into equivalence classes so that all the mentions in a class refer to the same entity
- Map pronouns to the respective proper nouns or other noun phrases.
- The paper uses Illinois coreference package
- We used Stanford CoreNLP Package

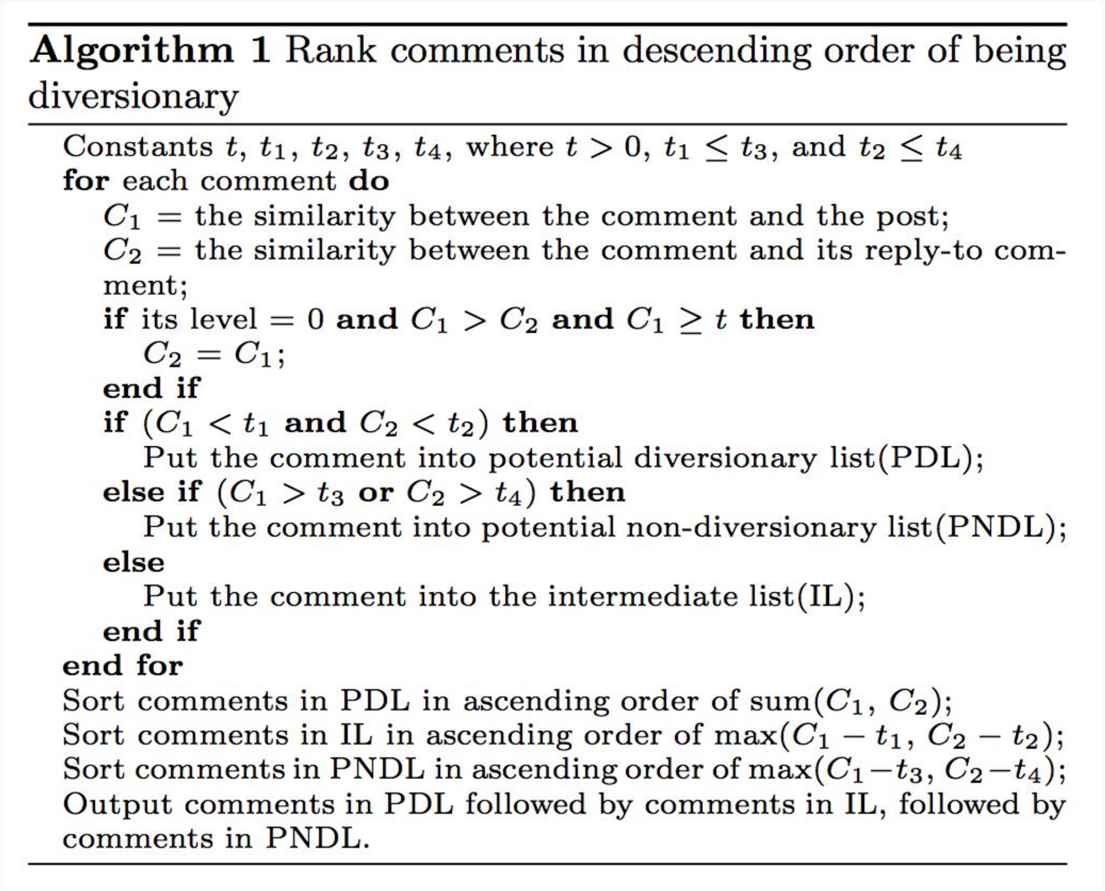
Extracting Proper Nouns

- Identify proper pronouns in the post
- Use all anchor tags present in first paragraph of their wiki page
- Adds hidden knowledge about entities
- PymediaWiki module of python

Topic modelling using LDA

- Given post and its comments have little data.
  Lots of data required!
- Query the title of the post and use the search result documents as training data.
  Build the training-LDA model
- Test data = comments of the post
- For each term in the test data
  - If term appeared in training data, use the term-topic distribution from training-LDA model
  - Else, apply equation (2) to get the term-topic distribution
- Using these term-topic distributions, obtain the document-topic distributions for each document (post and the comments) in the test data
- Using document-topic distributions as topic vectors, compute pairwise similarities

The Algorithm



Algorithm 1 Rank comments in descending order of being diversionary

```
Constants $t$, $t_1$, $t_2$, $t_3$, $t_4$, where $t > 0$, $t_1 \leq t_3$, and $t_2 \leq t_4$
for each comment do
    $C_1$ = the similarity between the comment and the post;
    $C_2$ = the similarity between the comment and its reply-to comment;
    if its level = 0 and $C_1 > C_2$ and $C_1 \geq t$ then
        $C_2 = C_1$;
    end if
    if ($C_1 < t_1$ and $C_2 < t_2$) then
        Put the comment into potential diversionary list(PDL);
    else if ($C_1 > t_3$ or $C_2 > t_4$) then
        Put the comment into potential non-diversionary list(PNDL);
    else
        Put the comment into the intermediate list(IL);
    end if
end for
Sort comments in PDL in ascending order of sum($C_1$, $C_2$);
Sort comments in IL in ascending order of max($C_1 - t_1$, $C_2 - t_2$);
Sort comments in PNDL in ascending order of max($C_1 - t_3$, $C_2 - t_4$);
Output comments in PDL followed by comments in IL, followed by comments in PNDL.
```

Threshold values:

- ➔ t = 50%
- ➔ t1 = 10%
- ➔ t2 = 20%
- ➔ t3 = 50%
- ➔ t4 = 90%

Link to github repo:

https://github.com/bhvjain/diversionary_comments.git