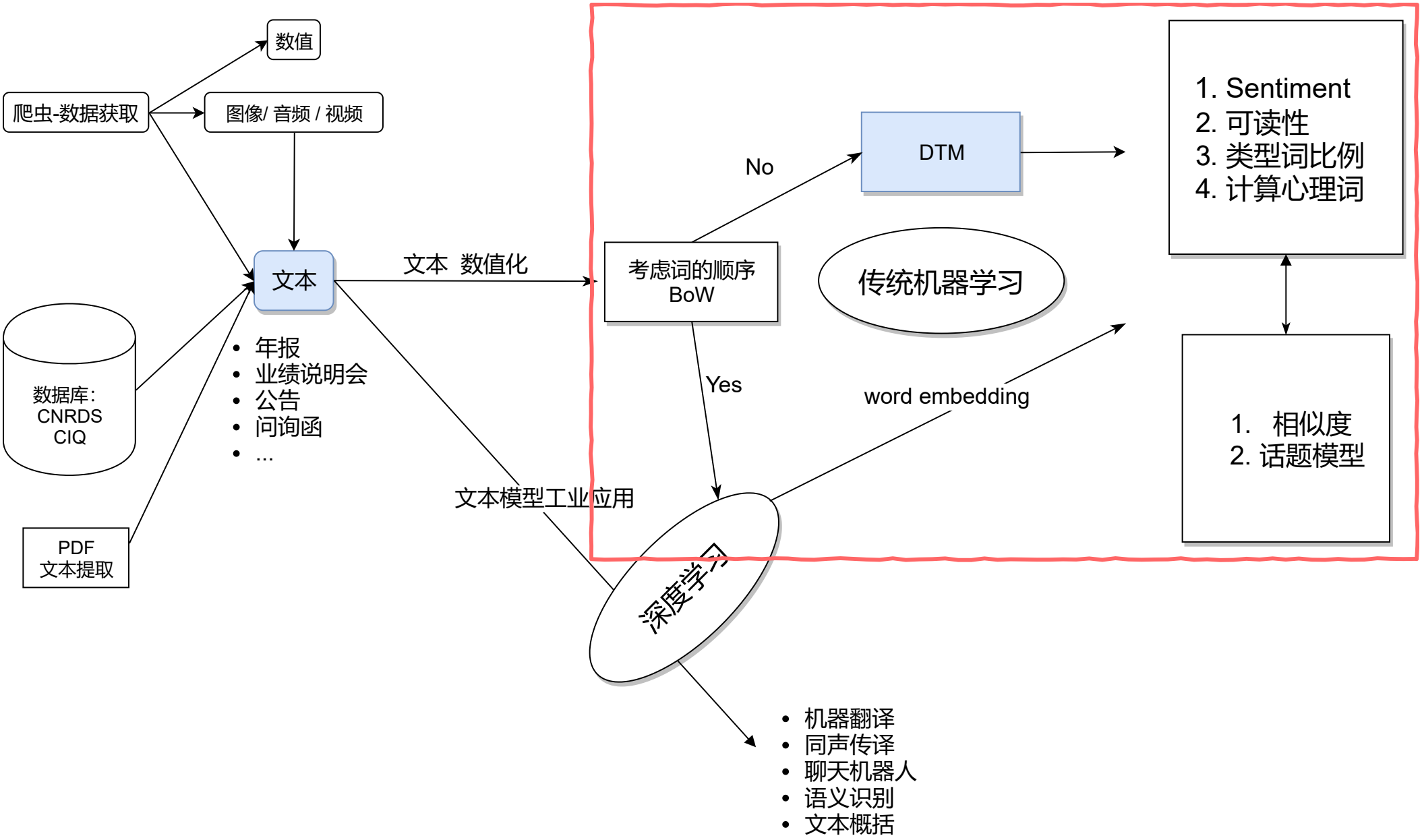
Text Analysis in Accounting Research

Methodology & Code

Xinlu Wang

16 MAY 2019

Framework


数值
爬虫-数据获取
图像/ 音频 / 视频
文本
文本 数值化
考虑词的顺序
BoW
No
DTM
传统机器学习
1. Sentiment
2. 可读性
3. 类型词比例
4. 计算心理词
数据库:
CNRDS
CIQ
年报
业绩说明会
公告
问询函
...
Yes
word embedding
1. 相似度
2. 话题模型
PDF
文本提取
文本模型工业应用
深度学习
机器翻译
同声传译
聊天机器人
语义识别
文本概括

实地调研

澄清公告

舞弊新闻

Methodology

Loughran and Mcdonald (2016)

Dictionary based Method

- Sentiment
- Readability / Fog Index

```
Fog Index = 0.4 (average number of words per sentence + percentage of complex words)
```

- specific list of words, e.g., "Competition", "I", "Non-GAAP", LIWC, etc.
- self developed word lists (Naive Bayes method)

Between Document

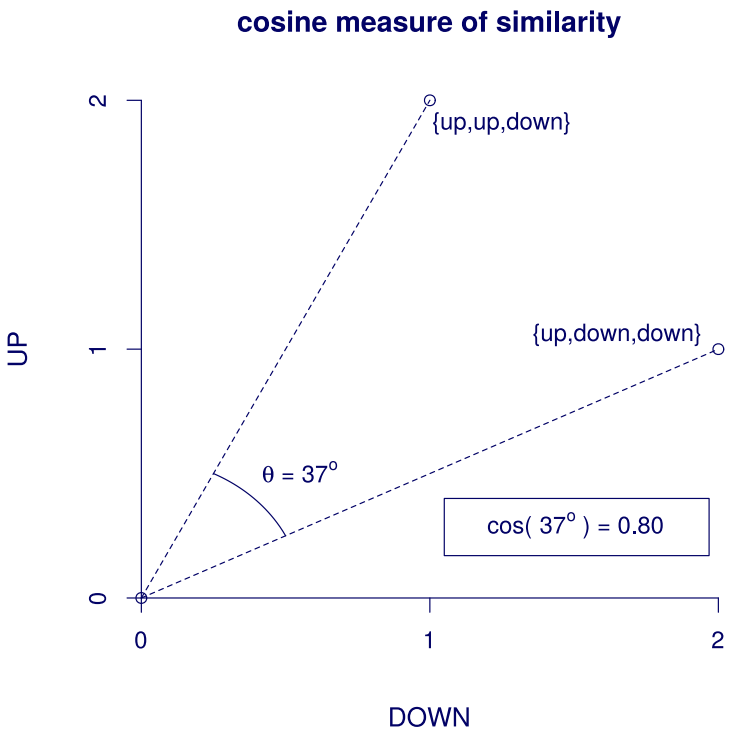
- Document similarity
- Topic Models (LSA, LDA)

Document Term Matrix (DTM)

	Term 1	Term 2	Term 3	Term 4	Term 5	Term 6	Term *n*
Doc 1	0	0	0	0	0	3	0
Doc 2	2	0	9	8	7	3	1
Doc 3	49	39	28	73	64	100	92
Doc 4	0	0	1938	27362	2737	1162	283
Doc 5		And so on...					
Doc 6							
Doc *n*							

- Stop words
- Stemming vs Lemmatizing
- Bi-gram
- TF-IDF : $tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$
- Corpus trimming

Similarity



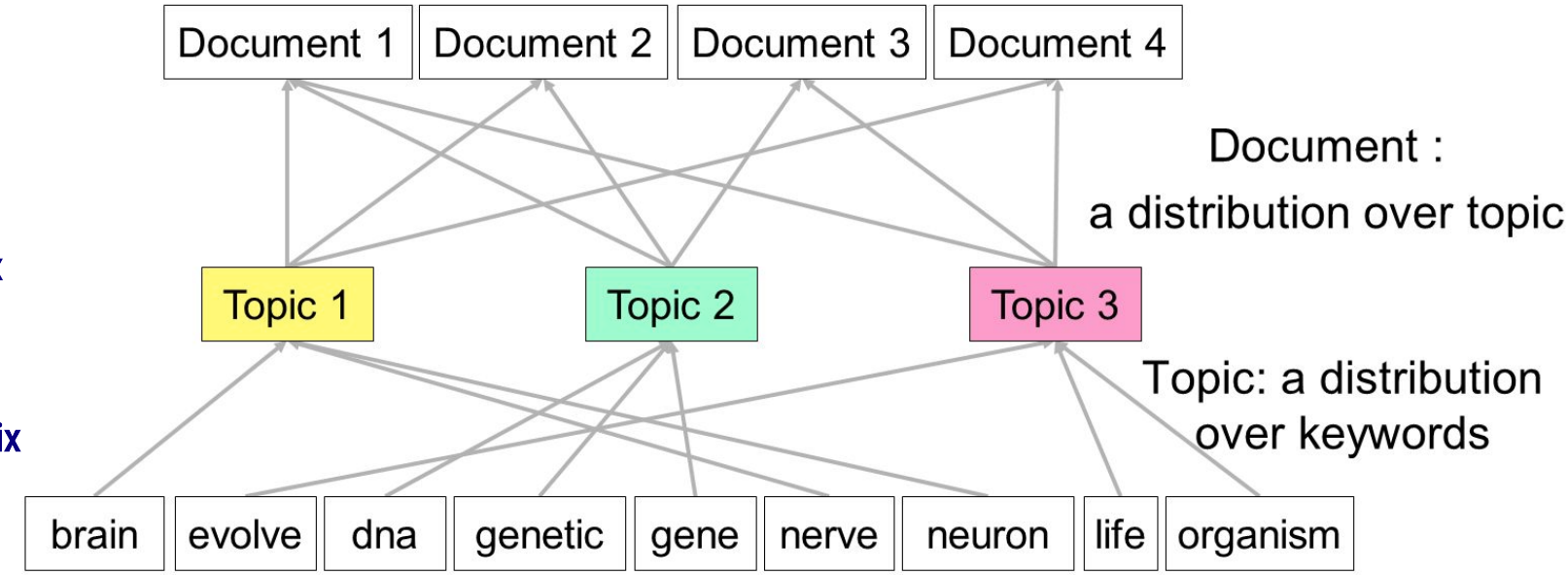
$$\text{Similarity}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| B\|}$$

- Document as a vector
- [0, 1]
- ignore length

topic models (Latent Dirichlet Allocation)

Output:

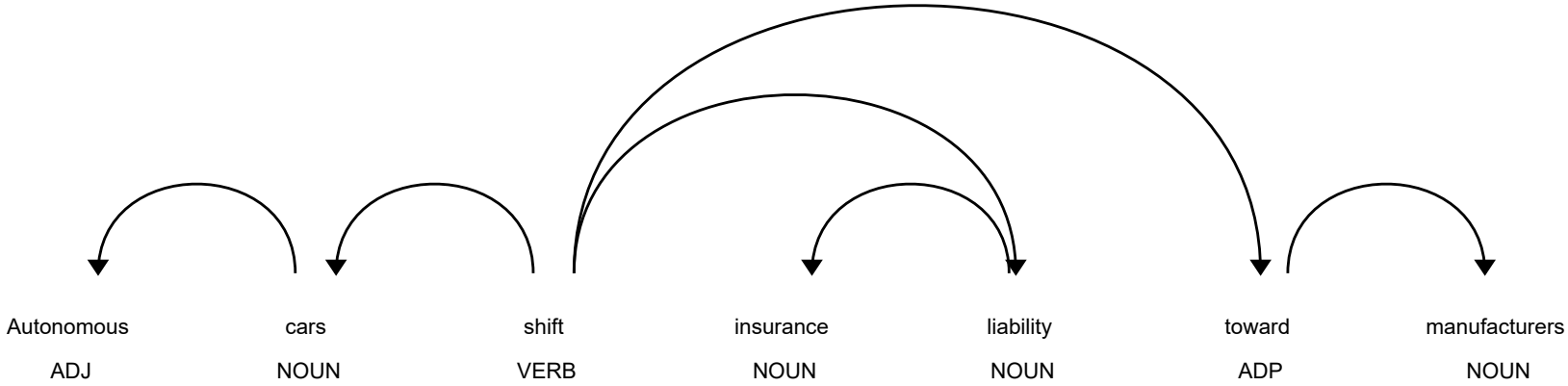
- Doc-Topic Matrix
- Topic-Term Matrix



Other Applications

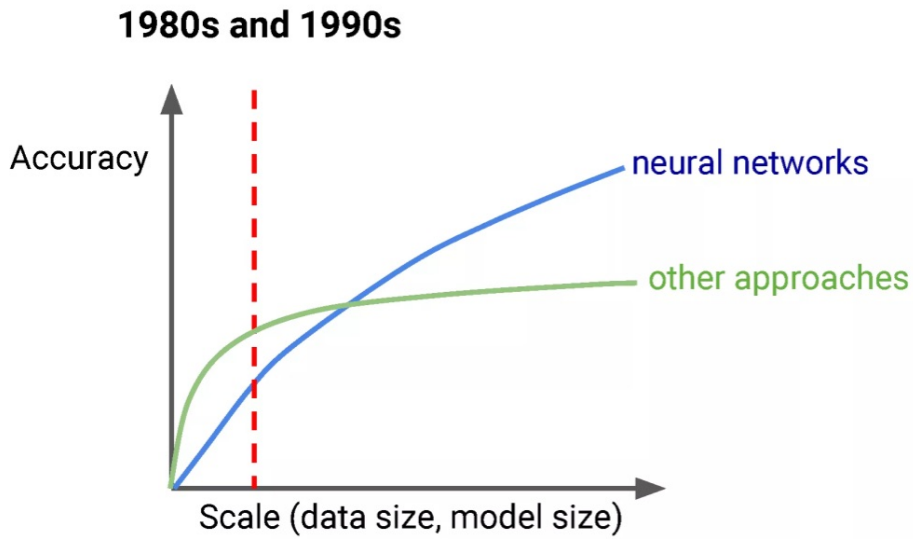
- SVR, e.g. Frankel, Jennings, and Lee (2016)
    - auto ML
- Clustering
- Classification
- STM, CTM, BTM e.g. Cheng, Yan, Lan, and Guo (2014)
- Network

Position of Speech Tagging



- **Named Entity Recognition**
- **Dependency**
- **Time Terms** e.g. Thorstad and Wolff (2018)

Deep Learning in NLP



- word2vec [CBoW or Skip-gram] etc.
  - King - man = Queen - Woman
  - Doctor - man + Woman = ?
- doc2vec, LDA2vec, etc.
- RNN, LSTM, etc.
- 预训练模型 BERT, ERNIE

> a word, a sentence, or a document ⇒ a vector
>
> Cloud API

Coding Tips

Python Ecosystem

- Anaconda
- `jupyter notebook`
  - nteract
- `pandas` and `numpy`
- `sklearn` DT Matrix / Similarity / LDA
- `gensim` Topic models / LDA
- `sqlite3` with Sqlite3 DB Browser

R Ecosystem

- R
- Microsoft Open R (MKL for optimal performance)
- Rtools (Windows only)
- Rstudio
- `tidyverse`
- `quanteda`
- `tidytext`
- `stm`

Rstudio + reticulate

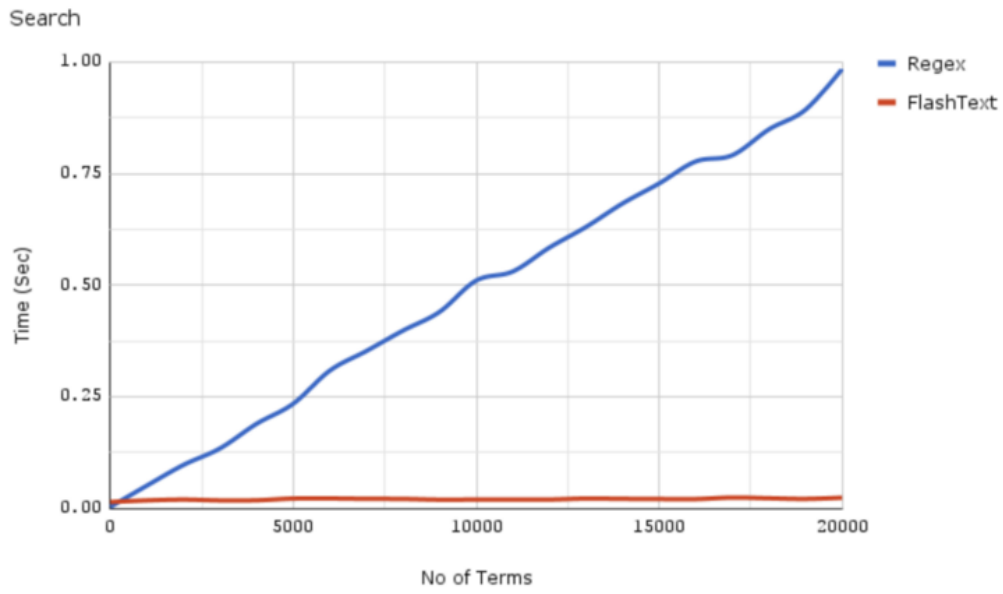
Readable regular expressions

```
from cursive_re import *

hash = text('#')
hexdigit = any_of(in_range('0', '9') + in_range('a', 'f') + in_range('A', 'F'))
hexcolor = (
    beginning_of_line() + hash +
    group(repeated(hexdigit, exactly=6) | repeated(hexdigit, exactly=3)) +
    end_of_line()
    )

str(hexcolor)
```

```
## '^\\#((?:[0-9a-fA-F]{6})|(?:[0-9a-fA-F]{3}))$'
```

flashtext

```
from flashtext import KeywordProcessor
keyword_processor = KeywordProcessor()
keyword_processor.add_keyword('Bay Area')
keyword_processor.extract_keywords('I love big Apple and Bay Area.')
# ['Bay Area']
```



分词

jieba

- 自定义词列表

```
import jieba
jieba.load_userdict("userdic.txt")
file["textclean"] = file.text.str.replace("\n", "". \
            apply(lambda x: " ".join(jieba.cut(x, cut_all=False)))
```

Stanford Model

```
cd /Users/Sean/nuts/Standford-NLP/stanford-segmenter-2017-06-09

./segment.sh pku textdf.txt UTF-8 0 > textdfseg.txt
```

剔除特殊字符 zhon

```
import zhon.hanzi as hanzi
import zhon.cedict as cedict
import re

def zhongonly(string):
   string = re.sub('[^{}]'.format(hanzi.characters), " ", string)
   string = ' '.join(string.split())
   return string

# hanzi.characters 只有汉字
# cedict.simplified 有顿号
# cedict.simplified + hanzi.punctuation 保留了标点符号
# 顿号 逗号 句号 引号 括号 书名号 分号 冒号

file["textcleanhan"]= file["textclean"].apply(zhongonly)
```

DTM

R

```
library(quanteda)
```

python

```
documents = item1a.item1Aclean.tolist() #

from sklearn.feature_extraction.text import CountVectorizer
#define vectorizer parameters
tfidf_vectorizer = CountVectorizer(max_df=0.99, max_features=10000,
                                   min_df=0.01,
                                   stop_words = stopwords, # clean stopwords
                                   analyzer='word')
tfidfc = tfidf_vectorizer.fit_transform(documents)
```

DTM TFIDF

```
documents = item1a.item1Aclean.tolist() #
# print(documents[0])

from sklearn.feature_extraction.text import TfidfVectorizer
#define vectorizer parameters
tfidf_vectorizer = TfidfVectorizer(max_df=0.99, max_features=10000,
                                   min_df=0.01,
                                   stop_words = stopwords, # clean stopwords
                                   use_idf=True, # whether to use IDF
                                   analyzer='word')
tfidf = tfidf_vectorizer.fit_transform(documents)

# similarity
def cos_cdist(X):
    x = tfidf[X["row"],] * tfidf[X["col"],].T
    return list(x.data)[0]

itemTA["SimiScore"] = itemTA[['row','col']].apply(cos_cdist, axis=1)
```

???

similarity matrix. 计算时间长

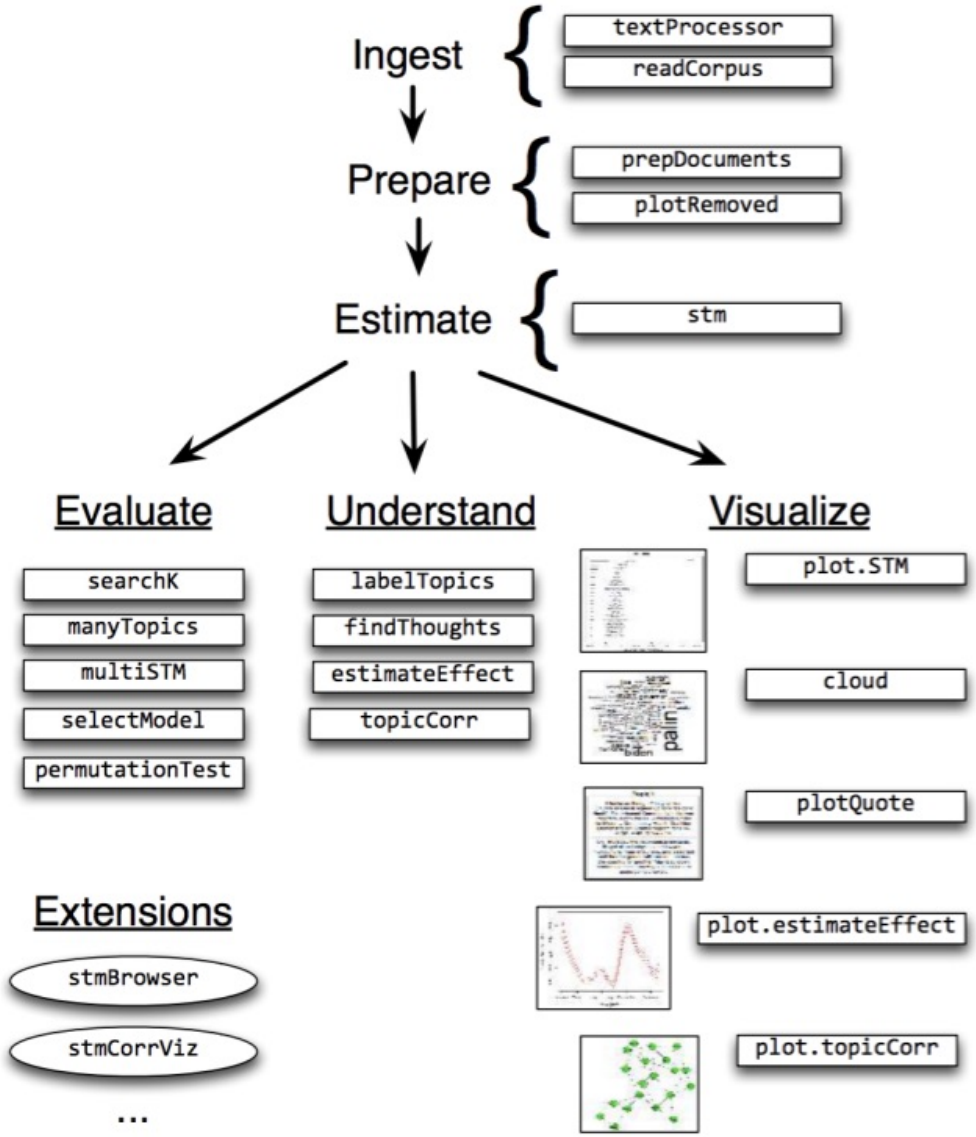
similarity function

structure topic models

```
library(stm)
library(quanteda)
library(tictoc)
library(feather)
library(tidyverse)

my_dfm <- readRDS("rf-feather/stm-data/my_dfm_all.RDS")

my_dfm_95 <- dfm_trim(my_dfm,
                      max_docfreq = 0.95,
                      min_docfreq = 0.001)

tic("train the model")
fitedmodel <- stm(my_dfm_95,
                  K = 100,  ## chanage
                  max.em.its = 75,
                  prevalence = ~  sic_1d + s(fyear))
toc()
```

structure topic models multiprocess

```
## multiprocess furrr map
library(furrr)
plan(multiprocess)

options(future.globals.maxSize= 10485760000)

tic("train the model")
many_models <- data_frame(K = c(30, 50, 75, 100, 150)) %>%
  mutate(topic_model = future_map(K,
                                  ~stm(my_dfm_95,
                                       K = . ,  ## chanage
                                       max.em.its = 75,
                                       prevalence = ~ factor(sic_1d))))

saveRDS(many_models, "rf-feather/stm-manymodels-sic1d.RDS", compress = TRUE)

toc()
```


Ingest
textProcessor
readCorpus
Prepare
prepDocuments
plotRemoved
Estimate
stm
Evaluate
searchK
manyTopics
multiSTM
selectModel
permutationTest
Understand
labelTopics
findThoughts
estimateEffect
topicCorr
Visualize
plot.STM
cloud
plotQuote
plot.estimateEffect
plot.topicCorr
Extensions
stmBrowser
stmCorrViz
...

Cloud API


Bai du 百度 | AI开放平台
产品服务
解决方案
合作伙伴
开发资源
AI加速器
语音技术
图像技术
文字识别
人脸与人体识别
视频技术
AR与VR
自然语言处理
数据智能
知识图谱
语言处理基础技术
词法分析
词向量表示
词义相似度
依存句法分析
DNN语言模型
短文本相似度 热门
文本纠错 新品
情感倾向分析 热门
评论观点抽取
对话情绪识别 新品
文章标签
文章分类
新闻摘要 邀测
文本审核 新品
机器翻译
通用翻译API
定制化翻译API
语音翻译SDK 新品
拍照翻译SDK 邀测

```
from urllib.request import urlopen
import urllib.request
import ssl

apikey = "fybyYDajn9pItPIoor307h"
secretkey = "MtQKLPlTMWcfaGcm5QxN1BjNyAOuDKG"
# client_id 为官网获取的AK， client_secret 为官网获取的SK
host = 'https://aip.baidubce.com/oauth/2.0/token?grant_type=client_credentials&client
            + apikey + '&client_secret=' + secretkey
request = urllib.request.Request(host)
request.add_header('Content-Type', 'application/json; charset=UTF-8')
response = urlopen(request)
content = response.read().decode()
print(content)
```

```
tokens = "24.4bf271fc8b1629f9ef62032ed0565f65.2592000.1541250309.282335-14348962"

urlpost = "https://aip.baidubce.com/rpc/2.0/nlp/v1/sentiment_classify"+ \
    "?access_token=" + tokens

import requests
import json

element = "苹果是一家伟大的公司"
params={'text':element}
encoded_data = json.dumps(params).encode('GBK')
r = requests.post(urlpost, data=encoded_data,
                  headers = {"Content-Type": "application/json"})

print(r.text)
```

```
# {
#     "text":"苹果是一家伟大的公司",
#     "items":[
#         {
#             "sentiment":2,    //表示情感极性分类结果
#             "confidence":0.40, //表示分类的置信度
#             "positive_prob":0.73, //表示属于积极类别的概率
#             "negative_prob":0.27  //表示属于消极类别的概率
#         }
#     ]
# }
```

References

Cheng, X, X. Yan, Y. Lan, et al. (2014). "BTM: Topic Modeling over Short Texts". In: *IEEE Transactions on Knowledge and Data Engineering* 26.12, pp. 2928-2941.

Frankel, R, J. Jennings, and J. Lee (2016). "Using unstructured and qualitative disclosures to explain accruals". In: *Journal of Accounting and Economics* 62.2, pp. 209-227.

Loughran, T. and B. Mcdonald (2016). "Textual Analysis in Accounting and Finance: A Survey". In: *Journal of Accounting Research* 54.4, pp. 1187-1230.

Thorstad, R. and P. Wolff (2018). "A big data analysis of the relationship between future thinking and decision-making". In: *Proceedings of the National Academy of Sciences* 115.8, pp. E1740-E1748.