## 训练阶段

* 用warplda处理训练数据
* 将生成的train.info.words.txt（每一行对应一个topic）看做一个文章集合，用article2vector工具转换成矩阵。
  + word2vec -train in.txt -output wordvec.txt -cbow 1 -size 200 -window 10 -negative 10 -hs 0 -sample 1e-5 -threads 8 -binary 0 -iter 3 -min-count 5
  + GLOG\_logtostderr=1 ./demo -algname jieba -algmgr localhost:9001 -port 10080 -vec wordvec -vecdict data/wordvec.txt -idx data/text\_class.annIdx
  + service jieba article2vector data/in.txt data/articlevec.txt
  + cd $api\_server/data && cat articlevec.txt | sort -k1,1n | awk '{$1=""; print $0}' > train.articlevec
  + cd algorithms/cppjieba-master/clustering/knn\_article && GLOG\_logtostderr=1 ./article\_knn.bin -build -input train.articlevec -nfields 100 -ntrees 100 -idx train.annIdx
* 用上一步生成的矩阵建立annoy tree: train.annIdx

## 预测阶段

* 读入请求文本text，用article2vector转成向量text.vec
* 用上一步生成的text.vec在train.annIdx上找knn，即是这个文本的k个最大的topic概率分布。

## 新方法，用Model和Vocab

准备数据

* 准备train数据，去掉文章的tag 前2个单词  
  cat ydir\_train.txt | awk '{$1=""; $2=""; print $0}' > train.data
* 将train.data用article2vector转成矩阵
  + 启动topic server  
    GLOG\_logtostderr=1 ./demo -algname topic\_pred -algmgr localhost:9001 -port 10080 -vec warplda -warpldaModel data/train.model -warpldaVocab data/train.vocab -idx data/train.annIdx  
    其中-idx可以随便用一个annoy tree文件，这一步不起作用
  + apiserver 端运行：service topic\_pred req:doc2vec in:data/train.data out:data/train.vec
  + 去掉train.vec的id  
    cat train.vec | awk '{$1=""; print $0}' > train.vector
* 用train.vector建立annoy tree：train.annIdx  
  cd algorithms/cppjieba-master/clustering/knn\_article && GLOG\_logtostderr=1 ./article\_knn.bin -build -input train.vector -nfields 100 -ntrees 100 -idx train.annIdx

上线

* 启动topic server 用上一步得到的train.annIdx  
  GLOG\_logtostderr=1 ./demo -algname topic\_pred -algmgr localhost:9001 -port 10080 -vec warplda -warpldaModel data/train.model -warpldaVocab data/train.vocab -idx data/train.annIdx
* 按照.so代码说明使用，注意批处理文章数据不要docid