Files

root/

config.py: Configuration of the project (variables and hyper-parameters).

preprocessor.py: Preprocess the data set and create train/dev/test sets.

data/

Data files. Put the original data files under this directory.

intention/

Data files for the intention module.

retrieval/

Data files for the retrieval module.

model/

intention/

Models for the intention module.

retrieval/

Models for the retrieval module.

intention/

business.py: Train a fasttext model and use it for classifying the intention of the user's query.

retrieval/

word2vec.py: Train a word2vec model on the JD dataset.

hnsw_faiss: HNSW model implemented with Faiss.

hnsw_hnswlib: HNSW model implemented with hnswlib.

TO-DO list:

必备资料

论文：*Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs*

模块1: 意图识别

intention/business.py:

任务: 完成Intention类。

目标：实现意图识别的模块。其中各部分实现如下，

build_keyword函数：用于提取标注数据的关键词，关键词的来源有两个，

```
1. train.csv中的名词（可使用jieba切词并作词性标注）
```

2. ware.txt中的sku词汇

data_process函数：用于对原始数据集进行标注，生成训练fasttext的数据集。在原始数据集中，包含keyword的样本标注为1（表示业务相关查询），否则为0。

train函数：训练fasttext分类器。

test函数：在test.csv上验证模型并计算F1score。

predict函数：用训练好的分类器进行预测。

主要使用工具：fastText pypi: fasttext

测试：python business.py

模块2: Embedding - Word2Vec

retrieval/word2vec.py:

任务: 完成train_w2v函数。

目标：在京东对话数据集上训练一个word2vec模型。

主要使用工具：gensim的word2vec

测试：python word2vec.py

模块3: ANNS召回 - HNSW

retrieval/hnsw_faiss.py:

任务1: 完成wam函数。

WAM即Word Average Model，使用average word vectors的方式来得到句向量。

任务2: 完成HNSW类。

目标：使用Faiss来实现HNSW的模块。

完成load_data, build_hnsw, load_hnsw, search, evaluate几个函数，其中需要注意的是，评估的方式我们可以直接以索引中的向量作为query，搜索最近的1个candidate，根据结果是否为自身来得到recall@1.

主要使用工具：

Faiss: Facebook AI Similarity Search

Github repo: https://github.com/facebookresearch/faiss/tree/13a2d4ef8fcb4aa8b92718ef4b9cc211033e7318

HNSW demos: https://github.com/facebookresearch/faiss/blob/13a2d4ef8fcb4aa8b92718ef4b9cc211033e7318/benchs/bench_hnsw.py

Build index

```
# Dim: Embedding demension
# M, ef_construction: defined in the paper "Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs"
index = faiss.IndexHNSWFlat(dim, M)
index.hnsw.efConstruction = ef_construction
index.verbose = True  # to see progress
index.add(vecs) # vecs: a n2-by-d matrix with query vectors
```

Save index to file and load index from file

```
# save index
faiss.write_index(index, file)
# load index
index = faiss.read_index(file)
```

Search in the index

```
# vecs: a n2-by-d matrix with query vectors
# D: distance
# I: Indexes of returned candidates
# k: number of nearest candidates
D, I = index.search(vecs, k)
```

Evaluate the index

```
nq, d = vecs.shape
t0 = time.time()
D, I = index.search(vecs, k)
t1 = time.time()

missing_rate = (I == -1).sum() / float(nq*k)
recall_at_1 = (I == np.arange(nq)).sum() / float(nq*k)
print("\t %7.3f ms per query, R@1 %.4f, missing rate %.4f" % (
  (t1 - t0) * 1000.0 / nq, recall_at_1, missing_rate))
```

评估方式：将index中已存在的部分query拿来作为测试集，查询其自身后通过判断返回的第一个结果是否为自身来计算recall@1。

测试：python hnsw_faiss.py

retrieval/hnsw_hnswlib.py:

(Optional) 任务3: 完成HNSW类。

目标：使用hnswlib来实现HNSW的模块。

主要使用工具：hnswlib

测试：python hnsw_hnswlib.py