doc2vec

April 27, 2016

1 Classification using Distributed representations of sentences and documents

Testing paragraph vectors approach as proposed in [1].
Date: 22.04.2016

1.1 Context

Different language models lead to differente expressiveness in the feature space and thus alternatives were explored. This particular approach promised state-of-the-art results.

1.2 Experiment Rationale

The main goal of these experiments was to compare the performance of the approach in [1] with a simple bag-of-words model, especially given the rather small dataset.

1.3 Testing Procedure and Metrics

Using [4,5] a bag-of-words model was trained on the the labelled sentence dataset with a TF.IDF transformation applied to it and then classification was carried out by

1.4 Test Results

From manual judgement the word2vec mapping works exceptionally well to find relations between words, however simply adding up vectors to represent several words was an overly naive approach and does not work (Note: In retrospective this approach does not even make sense mathematically.)

1.5 Learnings

- Word2vec mapping has potential if the thesis scope will focus on NLP (natural language processing) since it is aware of the local context of words as opposed to a simple bag-of-words approach.
- Representing documents needs a more sophisticated approach.

1.6 References

1. Le QV, Mikolov T (2014) Distributed representations of sentences and documents. arXiv preprint arXiv:1405.4053
2. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed Representations of Words and Phrases and their Compositionality. In: Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ, ed., Advances in Neural Information Processing Systems 26. Curran Associates, Inc.. 3111–3119
3. Mikolov T (2012) Statistical Language Models Based on Neural Networks. Ph.D. dissertation. Ph. D. thesis, Brno University of Technology.

4. http://scikit-learn.org/stable/
5. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay É (October 2011) Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12:2825−2830
6. https://radimrehurek.com/gensim/
7. Řehůřek R, Sojka P (May 2010) Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. Valletta, Malta: ELRA. 45–50

2 Code

In [8]:
```
def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn
```

In [9]:
```
import pickle
import datetime
import logging
import time
import pandas as pd
import numpy as np
from random import shuffle
import multiprocessing


import sys
sys.path.append('../joblearn-experiments/triton-experiments/workdir/')

import joblearn.dataset
import joblearn.gridsearch
import joblearn.feat_extr
import joblearn.feat_trans
import joblearn.scoring
import joblearn.target_trans
import joblearn.estimation

import sklearn.cross_validation
import sklearn.feature_extraction
import sklearn.linear_model
import sklearn.neighbors

import gensim
from gensim import models
```

In [10]:
```
### Basic setup

FEAT_DIM = 200

# test set size
TEST_SIZE = 0.3

# timestamp at runtime
```

```
TIMESTAMP = str(int(time.time()))
EXP_NAME = "paragraph2vec"
```

In [11]:
```
### Dataset Initialization

df = pd.read_csv(
    "../joblearn-experiments/local-experiments/workdir/data/sentences_aggregated_50-249.csv")

# Use entries with label confidence over 0.6 and aren't test questions:
df_conf = df[df['0_label:confidence'] > 0.6]
df_conf = df_conf[df_conf['_golden'] == False]
df_conf = df_conf[['0_label', '0_label:confidence', '0-sentence',
         '0-context-after', '0-context-before']]

label_array = np.array(df_conf['0_label'])
le = sklearn.preprocessing.LabelEncoder()
le.fit(label_array)
data_Y = le.transform(label_array)
data_Y_labels = le.classes_


data_X = np.array(df_conf['0-sentence'])

### Train/Test Splits Setup

label_groupings = {}
data_splits = {}

# no grouping
label_groupings["none"] = joblearn.target_trans.LabelGrouping("No grouping",
                                                              data_Y,
                                                              data_Y_labels)
(X_train, X_test, Y_train, Y_test) = sklearn.cross_validation.train_test_split(
     data_X, data_Y, test_size=TEST_SIZE, random_state=0)
data_splits["none"] = joblearn.target_trans.DataSplit(X_train, X_test,
                                                      Y_train, Y_test)
```

2.1 Doc2Vec

In [12]:
```
alldocs = []
for line_no, document in enumerate(data_X):
    words = gensim.utils.to_unicode(document).split()
    alldocs.append(gensim.models.doc2vec.LabeledSentence(words, [line_no]))

Xdocs = alldocs[:]
```

In [13]:
```
# model = Doc2Vec(documents, size=100, window=8, min_count=5, workers=4)
# model = gensim.models.Doc2Vec.load_word2vec_format(
#     '../05.1-tag-classification-tfidf-clustered/data/word2vec/text8.bin',
#     binary=True)  # C binary format

cores = multiprocessing.cpu_count()
assert gensim.models.doc2vec.FAST_VERSION > -1, "this will be painfully slow otherwise"
model = gensim.models.Doc2Vec(dm=1, dm_concat=1, size=100, window=10, negative=5, hs=0, min_co
```

In [14]:
```
model.build_vocab(Xdocs)
```

```
In [15]: for i in range(0,20):
             shuffle(Xdocs)
             model.train(Xdocs)
```

```
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller ‘batch_words’ for sm
```

```
In [16]: X_transformed_word2vec = np.matrix(model.docvecs)
```

2.2 N-grams BOW

```
In [17]: vectorizer = sklearn.feature_extraction.text.TfidfVectorizer(
             analyzer='word', max_features=100)
         X_transformed_bow = vectorizer.fit_transform(data_X).todense()
```

2.3 compare

```
In [18]: from sklearn import cross_validation
         from sklearn import linear_model
         from sklearn import neighbors
```

```
In [19]: (X_transformed_bow_train, X_transformed_bow_test,
          Y_bow_train, Y_bow_test) = cross_validation.train_test_split(
             X_transformed_bow, data_Y, test_size=0.3, random_state=0)

         (X_transformed_word2vec_train, X_transformed_word2vec_test, Y_word2vec_train,
          Y_word2vec_test) = cross_validation.train_test_split(
             X_transformed_word2vec, data_Y, test_size=0.3, random_state=0)
```

```
In [20]: # classifier_bow = linear_model.LogisticRegression().fit(X_transformed_bow_train, Y_bow_train)
         # classifier_word2vec = linear_model.LogisticRegression().fit(X_transformed_bow_train, Y_bow_t
         classifier_bow = neighbors.KNeighborsClassifier().fit(X_transformed_bow_train, Y_bow_train)
         classifier_word2vec = neighbors.KNeighborsClassifier().fit(X_transformed_bow_train, Y_bow_trai
```

```
In [21]: scores_word2vec = cross_validation.cross_val_score(
             classifier_word2vec, X_transformed_word2vec_test, Y_word2vec_test, cv=5,
             scoring='f1_weighted')

         print("Accuracy: %0.2f (+/- %0.2f)" % (scores_word2vec.mean(), scores_word2vec.std() * 2))
```

```
Accuracy: 0.34 (+/- 0.03)
```

```
In [22]: scores_bow = cross_validation.cross_val_score(
             classifier_bow, X_transformed_bow_test, Y_bow_test, cv=5,
             scoring='f1_weighted')

         print("Accuracy: %0.2f (+/- %0.2f)" % (scores_bow.mean(), scores_bow.std() * 2))
```

```
Accuracy: 0.63 (+/- 0.07)
```

3 Improved Doc2Vec

```
In [23]: model = gensim.models.Doc2Vec(dm=1, dm_concat=1, size=300, window=5, negative=5, hs=0, min_cou
         model.build_vocab(Xdocs)
         X_transformed_word2vec = np.matrix(model.docvecs)
```

```
In [24]: for i in range(0,20):
             shuffle(Xdocs)
             model.train(Xdocs)
```

```
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
WARNING:gensim.models.word2vec:under 10 jobs per worker: consider setting a smaller 'batch_words' for sm
```

```
In [30]: model.intersect_word2vec_format(
             '../05.1-tag-classification-tfidf-clustered/data/word2vec/GoogleNews-vectors-negative300.bi
             binary=True)
```

```
In [31]: (X_transformed_bow_train, X_transformed_bow_test,
          Y_bow_train, Y_bow_test) = cross_validation.train_test_split(
             X_transformed_bow, data_Y, test_size=0.3, random_state=0)

         (X_transformed_word2vec_train, X_transformed_word2vec_test, Y_word2vec_train,
          Y_word2vec_test) = cross_validation.train_test_split(
             X_transformed_word2vec, data_Y, test_size=0.3, random_state=0)
```

```
In [32]: classifier_bow = linear_model.LogisticRegression().fit(X_transformed_bow_train, Y_bow_train)
         classifier_word2vec = linear_model.LogisticRegression().fit(X_transformed_bow_train, Y_bow_tra
         # classifier_bow = neighbors.KNeighborsClassifier().fit(X_transformed_bow_train, Y_bow_train)
         # classifier_word2vec = neighbors.KNeighborsClassifier().fit(X_transformed_bow_train, Y_bow_tr
```

```
In [33]: scores_word2vec = cross_validation.cross_val_score(
             classifier_word2vec, X_transformed_word2vec_test, Y_word2vec_test, cv=5,
             scoring='f1_weighted')

         print("Accuracy: %0.2f (+/- %0.2f)" % (scores_word2vec.mean(), scores_word2vec.std() * 2))
```

```
Accuracy: 0.23 (+/- 0.00)
```

```
In [34]: scores_bow = cross_validation.cross_val_score(
             classifier_bow, X_transformed_bow_test, Y_bow_test, cv=5,
             scoring='f1_weighted')

         print("Accuracy: %0.2f (+/- %0.2f)" % (scores_bow.mean(), scores_bow.std() * 2))
```

```
Accuracy: 0.68 (+/- 0.06)
```

```
In [ ]:
```