Branch: develop

Find file | Copy path

gensim / docs / notebooks / FastText_Tutorial.ipynb

mpenkov Improve gensim documentation (numfocus) (#2591)

bcee414 on Oct 22, 2019

6 contributors

Raw | Blame | History

773 lines (773 sloc) 30.3 KB

Using FastText via Gensim

This tutorial is about using fastText (https://github.com/facebookresearch/fastText) model in Gensim. There are two ways you can use fastText in Gensim - Gensim's native implementation of fastText and Gensim wrapper for fastText's original C++ code. Here, we'll learn to work with fastText library for training word-embedding models, saving & loading them and performing similarity operations & vector lookups analogous to Word2Vec.

When to use FastText?

The main principle behind fastText is that the morphological structure of a word carries important information about the meaning of the word, which is not taken into account by traditional word embeddings, which train a unique word embedding for every individual word. This is especially significant for morphologically rich languages (German, Turkish) in which a single word can have a large number of morphological forms, each of which might occur rarely, thus making it hard to train good word embeddings.
fastText attempts to solve this by treating each word as the aggregation of its subwords. For the sake of simplicity and language-independence, subwords are taken to be the character ngrams of the word. The vector for a word is simply taken to be the sum of all vectors of its component char-ngrams.
According to a detailed comparison of Word2Vec and FastText in this notebook (Word2Vec_FastText_Comparison.ipynb), fastText does significantly better on syntactic tasks as compared to the original Word2Vec, especially when the size of the training corpus is small. Word2Vec slightly outperforms FastText on semantic tasks though. The differences grow smaller as the size of training corpus increases.
Training time for fastText is significantly higher than the Gensim version of Word2Vec (`15min 42s` vs `6min 42s` on text8, 17 mil tokens, 5 epochs, and a vector size of 100).
fastText can be used to obtain vectors for out-of-vocabulary (OOV) words, by summing up vectors for its component char-ngrams, provided at least one of the char-ngrams was present in the training data.

Training models

For the following examples, we'll use the Lee Corpus (which you already have if you've installed gensim) for training our model.

For using the wrapper for fastText, you need to have fastText setup locally to be able to train models. See installation instructions for fastText (https://github.com/facebookresearch/fastText/#requirements) if you don't have fastText installed already.

Using Gensim's implementation of fastText

In [1]:
```
from gensim.models.fasttext import FastText as FT_gensim
from gensim.test.utils import datapath

# Set file names for train and test data
corpus_file = datapath('lee_background.cor')

model_gensim = FT_gensim(size=100)

# build the vocabulary
model_gensim.build_vocab(corpus_file=corpus_file)

# train the model
model_gensim.train(
    corpus_file=corpus_file, epochs=model_gensim.epochs,
    total_examples=model_gensim.corpus_count, total_words=model_gensim.corpus_total_words
)

print(model_gensim)
```

```
FastText(vocab=1762, size=100, alpha=0.025)
```

Using wrapper for fastText's C++ code

In [2]:
```
from gensim.models.wrappers.fasttext import FastText as FT_wrapper

# Set FastText home to the path to the FastText executable
ft_home = '/home/misha/src/fastText-0.1.0/fasttext'

# train the model
model_wrapper = FT_wrapper.train(ft_home, corpus_file)

print(model_wrapper)
print(model_wrapper)
```

```
FastText(vocab=1763, size=100, alpha=0.025)
```

Training hyperparameters

Hyperparameters for training the model follow the same pattern as Word2Vec. FastText supports the following parameters from the original word2vec -

```
- model: Training architecture. Allowed values: `cbow`, `skipgram` (Default `cbow`)
- size: Size of embeddings to be learnt (Default 100)
- alpha: Initial learning rate (Default 0.025)
- window: Context window size (Default 5)
- min_count: Ignore words with number of occurrences below this (Default 5)
- loss: Training objective. Allowed values: `ns`, `hs`, `softmax` (Default `ns`)
- sample: Threshold for downsampling higher-frequency words (Default 0.001)
- negative: Number of negative words to sample, for `ns` (Default 5)
- iter: Number of epochs (Default 5)
- sorted_vocab: Sort vocab by descending frequency (Default 1)
- threads: Number of threads to use (Default 12)
```

In addition, FastText has three additional parameters -

```
- min_n: min length of char ngrams (Default 3)
- max_n: max length of char ngrams (Default 6)
- bucket: number of buckets used for hashing ngrams (Default 2000000)
```

Parameters `min_n` and `max_n` control the lengths of character ngrams that each word is broken down into while training and looking up embeddings. If `max_n` is set to 0, or to be lesser than `min_n`, no character ngrams are used, and the model effectively reduces to Word2Vec.

To bound the memory requirements of the model being trained, a hashing function is used that maps ngrams to integers in 1 to K. For hashing these character sequences, the Fowler-Noll-Vo hashing function (http://www.isthe.com/chongo/tech/comp/fnv) (FNV-1a variant) is employed.

Note: As in the case of Word2Vec, you can continue to train your model while using Gensim's native implementation of fastText.

Saving/loading models

Models can be saved and loaded via the `load` and `save` methods.

In [3]:
```
# saving a model trained via Gensim's fastText implementation
model_gensim.save('saved_model_gensim')
loaded_model = FT_gensim.load('saved_model_gensim')
print(loaded_model)

# saving a model trained via fastText wrapper
model_wrapper.save('saved_model_wrapper')
loaded_model = FT_wrapper.load('saved_model_wrapper')
print(loaded_model)
```

```
FastText(vocab=1762, size=100, alpha=0.025)
FastText(vocab=1763, size=100, alpha=0.025)
```

The `save_word2vec_method` causes the vectors for ngrams to be lost. As a result, a model loaded in this way will behave as a regular word2vec model.

Word vector lookup

Note: Operations like word vector lookups and similarity queries can be performed in exactly the same manner for both the implementations of fastText so they have been demonstrated using only the fastText wrapper here.

FastText models support vector lookups for out-of-vocabulary words by summing up character ngrams belonging to the word.

In [4]:
```
print('night' in model_wrapper.wv.vocab)
print('nights' in model_wrapper.wv.vocab)
print(model_wrapper['night'])
print(model_wrapper['nights'])
```

```
True
False
[ 0.6988306   0.7962038   0.04964953  0.11940945  0.45962736  0.02930021
 -0.148752    0.06151627 -0.09016804  0.28291386  0.47263005 -0.07578029
 -0.6577542  -0.27169365 -0.16403204 -0.01813792  0.16956581 -0.4356721
 -0.5964832   0.736336   -0.30984706  0.3427041   0.15378788 -0.5059883
 -0.15675616 -0.5815964  -0.07495894 -0.41970676  0.36809948 -0.26071918
 -0.46826273  0.15298584 -0.04599814  0.6263372   0.52292955 -0.02519639
  0.65333563 -0.1163021   0.22651657  0.3189898  -0.07596255 -0.22332282
  0.15195839 -0.39055198 -0.19732887 -0.2747241   0.04130132 -0.63913506
  0.18559387 -0.15314367  0.26563224  0.71324116 -0.04440507  0.05624504
 -0.11758088 -0.3253568  -0.22081989  0.6120215  -0.35792083 -0.333798
 -0.11835586 -0.0503935   0.3735655   0.49913588 -0.02427495  0.17332387
 -0.6308407  -0.3670084  -0.23201697  0.1555578  -0.3275684   0.0828054
 -0.01972355 -0.27277097 -0.25400817 -0.50337344  0.12651777  0.01878418
  0.21467368 -0.30219504  0.72938025  1.2315444   0.34624976 -0.7114608
  0.36338523  0.06543703  0.01345754 -0.15920149  0.13876723 -0.5582751
  0.38154316  0.18617174  0.4476739  -0.02872563  0.11876874 -0.02085596
 -0.64908224  0.03067067  0.14303452  0.33201975]
[ 0.6146148   0.70105475  0.04316702  0.10669094  0.4011963   0.02504568
 -0.13186908  0.05575202 -0.07716817  0.24856749  0.41678062 -0.06665172
 -0.5781625  -0.2382541  -0.14530264 -0.01657776  0.1496157  -0.38340995
 -0.52317756  0.64602286 -0.27437162  0.30193132  0.13466597 -0.4432936
 -0.13953276 -0.51243937 -0.06671739 -0.36839843  0.323204   -0.23012711
 -0.4134057   0.13342045 -0.03989897  0.5513306   0.46034276 -0.02355763
  0.5749811  -0.10196284  0.19741252  0.28229755 -0.06662108 -0.19657284
  0.1323219  -0.34543604 -0.17333041 -0.24169934  0.03771086 -0.563315
  0.16434433 -0.13390124  0.2337911   0.6275974  -0.03961363  0.04971414
 -0.10379436 -0.28675792 -0.19387211  0.5369245  -0.3134195  -0.29489765
 -0.10219254 -0.04510017  0.32892984  0.43901965 -0.02051542  0.15215051
 -0.5549378  -0.32053903 -0.20249408  0.1375998  -0.28648633  0.07105823
 -0.01473362 -0.24162163 -0.2248782  -0.44446456  0.11056102  0.01639792
  0.1877773  -0.2670066   0.6425655   1.083377    0.30128938 -0.6256704
  0.32030573  0.05688732  0.00961584 -0.1400448   0.12174114 -0.49109733
  0.3353805   0.16405603  0.39405888 -0.02574553  0.10243315 -0.0189576
 -0.5711417   0.02725987  0.12675735  0.29079968]
```

The word vector lookup operation only works if at least one of the component character ngrams is present in the training corpus. For example -

In [5]:
```
# Raises a KeyError since none of the character ngrams of the word `axe` are present in the training data
try:
    model_wrapper['axe']
except KeyError:
    #
    # trap the error here so it does not interfere
    # with the execution of the cells below
    #
    pass
else:
    assert False, 'the above code should have raised a KeyError'
```

The `in` operation works slightly differently from the original word2vec. It tests whether a vector for the given word exists or not, not whether the word is present in the word vocabulary. To test whether a word is present in the training word vocabulary -

In [6]:
```
# Tests if word present in vocab
print("word" in model_wrapper.wv.vocab)
# Tests if vector present for word
print("word" in model_wrapper)
```

```
False
True
```

Similarity operations

Similarity operations work the same way as word2vec. **Out-of-vocabulary words can also be used, provided they have at least one character ngram present in the training data.**

In [7]:
```
print("nights" in model_wrapper.wv.vocab)
print("night" in model_wrapper.wv.vocab)
```

```
model_wrapper.similarity("night", "nights")
```

```
False
True
```

```
Out[7]: 0.9999938
```

Syntactically similar words generally have high similarity in fastText models, since a large number of the component char-ngrams will be the same. As a result, fastText generally does better at syntactic tasks than Word2Vec. A detailed comparison is provided here (Word2Vec_FastText_Comparison.ipynb).

Other similarity operations

```
In [8]: # The example training corpus is a toy corpus, results are not expected to be good, for proof-of-con
        cept only
        model_wrapper.most_similar("nights")
```

```
Out[8]: [('night', 0.9999542236328125),
         ('flights', 0.9999502897262573),
         ('rights', 0.9999481439590454),
         ('night.', 0.9999478459358215),
         ('night,', 0.999945878982544),
         ('eight', 0.9999404549598694),
         ('quarter', 0.9999394416809082),
         ('hearing', 0.9999383091926575),
         ('light', 0.9999381303787231),
         ('during', 0.9999378323554993)]
```

```
In [9]: model_wrapper.n_similarity(['sushi', 'shop'], ['japanese', 'restaurant'])
```

```
Out[9]: 0.99997056
```

```
In [10]: model_wrapper.doesnt_match("breakfast cereal dinner lunch".split())
```

```
Out[10]: 'dinner'
```

```
In [11]: model_wrapper.most_similar(positive=['baghdad', 'england'], negative=['london'])
```

```
Out[11]: [('side', 0.999753475189209),
          ('inside', 0.9997509717941284),
          ('suicide', 0.9997484683990479),
          ('administrators', 0.9997476935386658),
          ('administration', 0.9997475743293762),
          ('Alliance', 0.9997474551200867),
          ('Three', 0.9997437000274658),
          ('Police', 0.9997435212135315),
          ('Minister,', 0.9997434616088867),
          ('end', 0.9997432827949524)]
```

```
In [12]: model_wrapper.accuracy(questions=datapath('questions-words.txt'))
```

```
Out[12]: [{'section': 'capital-common-countries', 'correct': [], 'incorrect': []},
          {'section': 'capital-world', 'correct': [], 'incorrect': []},
          {'section': 'currency', 'correct': [], 'incorrect': []},
          {'section': 'city-in-state', 'correct': [], 'incorrect': []},
          {'section': 'family',
           'correct': [],
           'incorrect': [('HE', 'SHE', 'HIS', 'HER'), ('HIS', 'HER', 'HE', 'SHE')]},
          {'section': 'gram1-adjective-to-adverb', 'correct': [], 'incorrect': []},
          {'section': 'gram2-opposite', 'correct': [], 'incorrect': []},
          {'section': 'gram3-comparative',
           'correct': [('GREAT', 'GREATER', 'LOW', 'LOWER'),
            ('LONG', 'LONGER', 'LOW', 'LOWER'),
            ('LOW', 'LOWER', 'GREAT', 'GREATER')],
           'incorrect': [('GOOD', 'BETTER', 'GREAT', 'GREATER'),
            ('GOOD', 'BETTER', 'LONG', 'LONGER'),
            ('GOOD', 'BETTER', 'LOW', 'LOWER'),
            ('GREAT', 'GREATER', 'LONG', 'LONGER'),
            ('GREAT', 'GREATER', 'GOOD', 'BETTER'),
            ('LONG', 'LONGER', 'GOOD', 'BETTER'),
            ('LONG', 'LONGER', 'GREAT', 'GREATER'),
            ('LOW', 'LOWER', 'GOOD', 'BETTER'),
            ('LOW', 'LOWER', 'LONG', 'LONGER')]},
          {'section': 'gram4-superlative',
           'correct': [('GOOD', 'BEST', 'GREAT', 'GREATEST'),
            ('GOOD', 'BEST', 'LARGE', 'LARGEST'),
            ('GOOD', 'BEST', 'BIG', 'BIGGEST'),
            ('GREAT', 'GREATEST', 'LARGE', 'LARGEST'),
            ('GREAT', 'GREATEST', 'BIG', 'BIGGEST'),
            ('LARGE', 'LARGEST', 'BIG', 'BIGGEST'),
            ('LARGE', 'LARGEST', 'GREAT', 'GREATEST')],
           'incorrect': [('BIG', 'BIGGEST', 'GOOD', 'BEST'),
            ('BIG', 'BIGGEST', 'GREAT', 'GREATEST'),
            ('BIG', 'BIGGEST', 'LARGE', 'LARGEST'),
            ('GREAT', 'GREATEST', 'GOOD', 'BEST'),
            ('LARGE', 'LARGEST', 'GOOD', 'BEST')]},
          {'section': 'gram5-present-participle',
           'correct': [('GO', 'GOING', 'LOOK', 'LOOKING'),
            ('GO', 'GOING', 'SAY', 'SAYING'),
            ('PLAY', 'PLAYING', 'SAY', 'SAYING'),
            ('PLAY', 'PLAYING', 'LOOK', 'LOOKING'),
            ('SAY', 'SAYING', 'LOOK', 'LOOKING'),
            ('SAY', 'SAYING', 'PLAY', 'PLAYING')],
           'incorrect': [('GO', 'GOING', 'PLAY', 'PLAYING'),
            ('GO', 'GOING', 'RUN', 'RUNNING'),
            ('LOOK', 'LOOKING', 'PLAY', 'PLAYING'),
            ('LOOK', 'LOOKING', 'RUN', 'RUNNING'),
            ('LOOK', 'LOOKING', 'SAY', 'SAYING'),
            ('LOOK', 'LOOKING', 'GO', 'GOING'),
            ('PLAY', 'PLAYING', 'RUN', 'RUNNING'),
            ('PLAY', 'PLAYING', 'GO', 'GOING'),
            ('RUN', 'RUNNING', 'SAY', 'SAYING'),
            ('RUN', 'RUNNING', 'GO', 'GOING'),
            ('RUN', 'RUNNING', 'LOOK', 'LOOKING'),
            ('RUN', 'RUNNING', 'PLAY', 'PLAYING'),
            ('SAY', 'SAYING', 'GO', 'GOING'),
            ('SAY', 'SAYING', 'RUN', 'RUNNING')]},
          {'section': 'gram6-nationality-adjective',
           'correct': [('AUSTRALIA', 'AUSTRALIAN', 'INDIA', 'INDIAN'),
            ('AUSTRALIA', 'AUSTRALIAN', 'ISRAEL', 'ISRAELI'),
            ('INDIA', 'INDIAN', 'AUSTRALIA', 'AUSTRALIAN')],
           'incorrect': [('AUSTRALIA', 'AUSTRALIAN', 'FRANCE', 'FRENCH'),
            ('AUSTRALIA', 'AUSTRALIAN', 'SWITZERLAND', 'SWISS'),
            ('FRANCE', 'FRENCH', 'INDIA', 'INDIAN'),
            ('FRANCE', 'FRENCH', 'ISRAEL', 'ISRAELI'),
            ('FRANCE', 'FRENCH', 'SWITZERLAND', 'SWISS'),
            ('FRANCE', 'FRENCH', 'AUSTRALIA', 'AUSTRALIAN'),
            ('INDIA', 'INDIAN', 'ISRAEL', 'ISRAELI'),
            ('INDIA', 'INDIAN', 'SWITZERLAND', 'SWISS'),
            ('INDIA', 'INDIAN', 'FRANCE', 'FRENCH'),
            ('ISRAEL', 'ISRAELI', 'SWITZERLAND', 'SWISS'),
            ('ISRAEL', 'ISRAELI', 'AUSTRALIA', 'AUSTRALIAN'),
            ('ISRAEL', 'ISRAELI', 'FRANCE', 'FRENCH'),
            ('ISRAEL', 'ISRAELI', 'INDIA', 'INDIAN'),
            ('SWITZERLAND', 'SWISS', 'AUSTRALIA', 'AUSTRALIAN'),
            ('SWITZERLAND', 'SWISS', 'FRANCE', 'FRENCH'),
            ('SWITZERLAND', 'SWISS', 'INDIA', 'INDIAN'),
            ('SWITZERLAND', 'SWISS', 'ISRAEL', 'ISRAELI')]},
          {'section': 'gram7-past-tense',
           'correct': [('PAYING', 'PAID', 'SAYING', 'SAID')],
           'incorrect': [('GOING', 'WENT', 'PAYING', 'PAID'),
            ('GOING', 'WENT', 'PLAYING', 'PLAYED'),
            ('GOING', 'WENT', 'SAYING', 'SAID'),
            ('GOING', 'WENT', 'TAKING', 'TOOK'),
            ('PAYING', 'PAID', 'PLAYING', 'PLAYED'),
            ('PAYING', 'PAID', 'TAKING', 'TOOK'),
            ('PAYING', 'PAID', 'GOING', 'WENT'),
            ('PLAYING', 'PLAYED', 'SAYING', 'SAID'),
            ('PLAYING', 'PLAYED', 'TAKING', 'TOOK'),
            ('PLAYING', 'PLAYED', 'GOING', 'WENT'),
            ('PLAYING', 'PLAYED', 'PAYING', 'PAID'),
            ('SAYING', 'SAID', 'TAKING', 'TOOK'),
            ('SAYING', 'SAID', 'GOING', 'WENT'),
            ('SAYING', 'SAID', 'PAYING', 'PAID'),
            ('SAYING', 'SAID', 'PLAYING', 'PLAYED'),
            ('TAKING', 'TOOK', 'GOING', 'WENT'),
            ('TAKING', 'TOOK', 'PAYING', 'PAID'),
            ('TAKING', 'TOOK', 'PLAYING', 'PLAYED'),
            ('TAKING', 'TOOK', 'SAYING', 'SAID')]},
          {'section': 'gram8-plural',
           'correct': [('MAN', 'MEN', 'CAR', 'CARS')],
           'incorrect': [('BUILDING', 'BUILDINGS', 'CAR', 'CARS'),
            ('BUILDING', 'BUILDINGS', 'CHILD', 'CHILDREN'),
            ('BUILDING', 'BUILDINGS', 'MAN', 'MEN'),
            ('CAR', 'CARS', 'CHILD', 'CHILDREN'),
            ('CAR', 'CARS', 'MAN', 'MEN'),
             ('CAR', 'CARS', 'BUILDING', 'BUILDINGS'),
             ('CHILD', 'CHILDREN', 'MAN', 'MEN'),
             ('CHILD', 'CHILDREN', 'BUILDING', 'BUILDINGS'),
             ('CHILD', 'CHILDREN', 'CAR', 'CARS'),
             ('MAN', 'MEN', 'BUILDING', 'BUILDINGS'),
             ('MAN', 'MEN', 'CHILD', 'CHILDREN')]},
           {'section': 'gram9-plural-verbs', 'correct': [], 'incorrect': []},
           {'section': 'total',
            'correct': [('GREAT', 'GREATER', 'LOW', 'LOWER'),
             ('LONG', 'LONGER', 'LOW', 'LOWER'),
             ('LOW', 'LOWER', 'GREAT', 'GREATER'),
             ('GOOD', 'BEST', 'GREAT', 'GREATEST'),
             ('GOOD', 'BEST', 'LARGE', 'LARGEST'),
             ('GOOD', 'BEST', 'BIG', 'BIGGEST'),
             ('GREAT', 'GREATEST', 'LARGE', 'LARGEST'),
             ('GREAT', 'GREATEST', 'BIG', 'BIGGEST'),
             ('LARGE', 'LARGEST', 'BIG', 'BIGGEST'),
             ('LARGE', 'LARGEST', 'GREAT', 'GREATEST'),
             ('GO', 'GOING', 'LOOK', 'LOOKING'),
             ('GO', 'GOING', 'SAY', 'SAYING'),
             ('PLAY', 'PLAYING', 'SAY', 'SAYING'),
             ('PLAY', 'PLAYING', 'LOOK', 'LOOKING'),
             ('SAY', 'SAYING', 'LOOK', 'LOOKING'),
             ('SAY', 'SAYING', 'PLAY', 'PLAYING'),
             ('AUSTRALIA', 'AUSTRALIAN', 'INDIA', 'INDIAN'),
             ('AUSTRALIA', 'AUSTRALIAN', 'ISRAEL', 'ISRAELI'),
             ('INDIA', 'INDIAN', 'AUSTRALIA', 'AUSTRALIAN'),
             ('PAYING', 'PAID', 'SAYING', 'SAID'),
             ('MAN', 'MEN', 'CAR', 'CARS')],
            'incorrect': [('HE', 'SHE', 'HIS', 'HER'),
             ('HIS', 'HER', 'HE', 'SHE'),
             ('GOOD', 'BETTER', 'GREAT', 'GREATER'),
             ('GOOD', 'BETTER', 'LONG', 'LONGER'),
             ('GOOD', 'BETTER', 'LOW', 'LOWER'),
             ('GREAT', 'GREATER', 'LONG', 'LONGER'),
             ('GREAT', 'GREATER', 'GOOD', 'BETTER'),
             ('LONG', 'LONGER', 'GOOD', 'BETTER'),
             ('LONG', 'LONGER', 'GREAT', 'GREATER'),
             ('LOW', 'LOWER', 'GOOD', 'BETTER'),
             ('LOW', 'LOWER', 'LONG', 'LONGER'),
             ('BIG', 'BIGGEST', 'GOOD', 'BEST'),
             ('BIG', 'BIGGEST', 'GREAT', 'GREATEST'),
             ('BIG', 'BIGGEST', 'LARGE', 'LARGEST'),
             ('GREAT', 'GREATEST', 'GOOD', 'BEST'),
             ('LARGE', 'LARGEST', 'GOOD', 'BEST'),
             ('GO', 'GOING', 'PLAY', 'PLAYING'),
             ('GO', 'GOING', 'RUN', 'RUNNING'),
             ('LOOK', 'LOOKING', 'PLAY', 'PLAYING'),
             ('LOOK', 'LOOKING', 'RUN', 'RUNNING'),
             ('LOOK', 'LOOKING', 'SAY', 'SAYING'),
             ('LOOK', 'LOOKING', 'GO', 'GOING'),
             ('PLAY', 'PLAYING', 'RUN', 'RUNNING'),
             ('PLAY', 'PLAYING', 'GO', 'GOING'),
             ('RUN', 'RUNNING', 'SAY', 'SAYING'),
             ('RUN', 'RUNNING', 'GO', 'GOING'),
             ('RUN', 'RUNNING', 'LOOK', 'LOOKING'),
             ('RUN', 'RUNNING', 'PLAY', 'PLAYING'),
             ('SAY', 'SAYING', 'GO', 'GOING'),
             ('SAY', 'SAYING', 'RUN', 'RUNNING'),
             ('AUSTRALIA', 'AUSTRALIAN', 'FRANCE', 'FRENCH'),
             ('AUSTRALIA', 'AUSTRALIAN', 'SWITZERLAND', 'SWISS'),
             ('FRANCE', 'FRENCH', 'INDIA', 'INDIAN'),
             ('FRANCE', 'FRENCH', 'ISRAEL', 'ISRAELI'),
             ('FRANCE', 'FRENCH', 'SWITZERLAND', 'SWISS'),
             ('FRANCE', 'FRENCH', 'AUSTRALIA', 'AUSTRALIAN'),
             ('INDIA', 'INDIAN', 'ISRAEL', 'ISRAELI'),
             ('INDIA', 'INDIAN', 'SWITZERLAND', 'SWISS'),
             ('INDIA', 'INDIAN', 'FRANCE', 'FRENCH'),
             ('ISRAEL', 'ISRAELI', 'SWITZERLAND', 'SWISS'),
             ('ISRAEL', 'ISRAELI', 'AUSTRALIA', 'AUSTRALIAN'),
             ('ISRAEL', 'ISRAELI', 'FRANCE', 'FRENCH'),
             ('ISRAEL', 'ISRAELI', 'INDIA', 'INDIAN'),
             ('SWITZERLAND', 'SWISS', 'AUSTRALIA', 'AUSTRALIAN'),
             ('SWITZERLAND', 'SWISS', 'FRANCE', 'FRENCH'),
             ('SWITZERLAND', 'SWISS', 'INDIA', 'INDIAN'),
             ('SWITZERLAND', 'SWISS', 'ISRAEL', 'ISRAELI'),
             ('GOING', 'WENT', 'PAYING', 'PAID'),
             ('GOING', 'WENT', 'PLAYING', 'PLAYED'),
             ('GOING', 'WENT', 'SAYING', 'SAID'),
             ('GOING', 'WENT', 'TAKING', 'TOOK'),
             ('PAYING', 'PAID', 'PLAYING', 'PLAYED'),
             ('PAYING', 'PAID', 'TAKING', 'TOOK'),
             ('PAYING', 'PAID', 'GOING', 'WENT'),
             ('PLAYING', 'PLAYED', 'SAYING', 'SAID'),
             ('PLAYING', 'PLAYED', 'TAKING', 'TOOK'),
             ('PLAYING', 'PLAYED', 'GOING', 'WENT'),
             ('PLAYING', 'PLAYED', 'PAYING', 'PAID'),
             ('SAYING', 'SAID', 'TAKING', 'TOOK'),
             ('SAYING', 'SAID', 'GOING', 'WENT'),
             ('SAYING', 'SAID', 'PAYING', 'PAID'),
             ('SAYING', 'SAID', 'PLAYING', 'PLAYED'),
             ('TAKING', 'TOOK', 'GOING', 'WENT'),
             ('TAKING', 'TOOK', 'PAYING', 'PAID'),
             ('TAKING', 'TOOK', 'PLAYING', 'PLAYED'),
             ('TAKING', 'TOOK', 'SAYING', 'SAID'),
             ('BUILDING', 'BUILDINGS', 'CAR', 'CARS'),
             ('BUILDING', 'BUILDINGS', 'CHILD', 'CHILDREN'),
             ('BUILDING', 'BUILDINGS', 'MAN', 'MEN'),
             ('CAR', 'CARS', 'CHILD', 'CHILDREN'),
             ('CAR', 'CARS', 'MAN', 'MEN'),
             ('CAR', 'CARS', 'BUILDING', 'BUILDINGS'),
```