```
import tensorflow as tf

import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from sklearn.model_selection import train_test_split
import pandas as pd
import unicodedata
import re
import numpy as np
import os
from os import path
import io
import time
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

from keras.models import Sequential
from keras.layers import Bidirectional, Concatenate, Permute, Dot, Input, GRU,LSTM, Multiply
from keras.layers import RepeatVector,TimeDistributed, Dense, Activation, Lambda,BatchNormalization
from keras.optimizers import Adam
from keras.utils import to_categorical
from keras.models import load_model, Model
import keras.backend as K
```

+ Code + Text

```
%%bash
wget http://opus.nlpl.eu/download.php?f=OpenSubtitles/v2018/mono/OpenSubtitles.raw.en.gz
mkdir /tmp/data/
gunzip -c download.php?f=OpenSubtitles%2Fv2018%2Fmono%2FOpenSubtitles.raw.en.gz > /tmp/lines
split -a 3 -l 100000  /tmp/lines /tmp/data/lines-
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
```

```
os.listdir('/tmp/data/')
```

```
['lines-fpv',
 'lines-arq',
 'lines-dqv',
 'lines-eru',
 'lines-bov',
 'lines-eby',
 'lines-cdr',
 'lines-cbl',
 'lines-fdl',
 'lines-bwr',
 'lines-dtm',
 'lines-etw',
 'lines-eqo',
 'lines-cht',
 'lines-amb',
 'lines-bdx',
 'lines-bdd',
 'lines-dfb',
 'lines-fax',
 'lines-ehn',
 'lines-cnb',
 'lines-fea',
 'lines-ftw',
 'lines-czo',
 'lines-gnm',
 'lines-cyu',
 'lines-avn',
 'lines-gaz',
 'lines-cfd',
 'lines-fbs',
 'lines-bpv',
 'lines-gkf',
 'lines-dle',
 'lines-flr',
 'lines-aom',
 'lines-dhy',
 'lines-ajj',
 'lines-cye',
 'lines-cco',
 'lines-cix',
 'lines-ghi',
 'lines-dxw',
 'lines-dab',
```

```
        'lines-fxx',
        'lines-dgn',
        'lines-gln',
        'lines-bes',
        'lines-fvp',
        'lines-epw',
        'lines-fkx',
        'lines-bxs',
        'lines-bcg',
        'lines-fpk',
        'lines-bqo',
        'lines-eme',
        'lines-fwg',
        'lines-dhh',
        'lines-aae',
        'lines-dci',
        'lines-esv',
        'lines-dlf',
        'lines-foz',
        'lines-dlw',
        'lines-bmg',
        'lines-cvh',
        'lines-fmt',
        'lines-adp',
        'lines-ftf',
        'lines-epo',
        'lines-gcs',
        'lines-ecv',
        'lines-aor',
        'lines-eez',
        'lines-gmj',
        'lines-dpk',
        'lines-aii',
        'lines-fbw',
        'lines-cko',
        'lines-apq',
        'lines-aex',
        'lines-chb',
        'lines-fpp',
        'lines-ggd',
        'lines-gfk',
        'lines-aic',
        'lines-dig',
        'lines-fmk',
```

```
          'lines-fac',
          'lines-ffp',
          'lines-clu',
          'lines-bad',
          'lines-dbu',
          'lines-cin',
          'lines-fjr',
          'lines-abf',
          'lines-bnb',
          'lines-ccm',
          'lines-aeh',
          'lines-cfv',
          'lines-ejy',
          'lines-agf',
          'lines-emt',
          'lines-bmk',
          'lines-ehw',
          'lines-dos',
          'lines-ftm',
          'lines-and',
          'lines-dyf',
          'lines-acg',
          'lines-fql',
          'lines-dsw',
          'lines-flc',
          'lines-bhm',
          'lines-bzd',
          'lines-cuq',
          'lines-gbw',
          'lines-ast',
          'lines-bsk',
          'lines-cgi',
          'lines-gad',
          'lines-akc',
          'lines-exi',
          'lines-cjr',
          'lines-fbc',
          'lines-fpd',
          'lines-gfh',
          'lines-crm',
          'lines-ejg',
          'lines-geo',
          'lines-ahz',
```

```
        'lines-dug',
        'lines-dyn',
        'lines-cqh',
        'lines-cef',
        'lines-dsx',
        'lines-cvr',
        'lines-abz',
        'lines-btt',
        'lines-bxf',
        'lines-fgs',
        'lines-cav',
        'lines-dyo',
        'lines-cry',
        'lines-ezs',
        'lines-dpa',
        'lines-fox',
        'lines-fyv',
        'lines-eus',
        'lines-duh',
        'lines-fnm',
        'lines-azs',
        'lines-cvq',
        'lines-cce',
        'lines-cvl',
        'lines-dje',
        'lines-dnm',
        'lines-ezt',
        'lines-cqd',
        'lines-dgi',
        'lines-eti',
        'lines-dhe',
        'lines-dpl',
        'lines-gld',
        'lines-afw',
        'lines-foe',
        'lines-fyh',
        'lines-fzi',
        'lines-akq',
        'lines-fzc',
        'lines-dhm',
        'lines-gdm',
        'lines-fhk',
        'lines-cqe',
        'lines-fpg',
```

```
        'lines-bom',
        'lines-gnn',
        'lines-cio',
        'lines-bnh',
        'lines-ggc',
        'lines-fmf',
        'lines-enu',
        'lines-coo',
        'lines-erj',
        'lines-fwb',
        'lines-bte',
        'lines-chv',
        'lines-ebq',
        'lines-eaa',
        'lines-aus',
        'lines-ddt',
        'lines-dbo',
        'lines-awu',
        'lines-eag',
        'lines-epr',
        'lines-adk',
        'lines-bum',
        'lines-afi',
        'lines-ejt',
        'lines-dny',
        'lines-gir',
        'lines-ahe',
        'lines-cqv',
        'lines-cka',
        'lines-ahh',
        'lines-fcv',
        'lines-ayv',
        'lines-bgs',
        'lines-bsf',
        'lines-bfe',
        'lines-cis',
        'lines-bmn',
        'lines-gfg',
        'lines-fog',
        'lines-byy',
        'lines-dpd',
        'lines-aib',
        'lines-gmg',
```

```
        'lines-gga',
        'lines-crv',
        'lines-bse',
        'lines-fsz',
        'lines-afe',
        'lines-eah',
        'lines-fat',
        'lines-dhk',
        'lines-brw',
        'lines-bup',
        'lines-drn',
        'lines-exz',
        'lines-ayq',
        'lines-fbj',
        'lines-eos',
        'lines-efu',
        'lines-bll',
        'lines-gfj',
        'lines-cxw',
        'lines-dkt',
        'lines-awj',
        'lines-aty',
        'lines-atj',
        'lines-dxx',
        'lines-aiv',
        'lines-fia',
        'lines-fdy',
        'lines-cjn',
        'lines-cfl',
        'lines-czn',
        'lines-bvp',
        'lines-ass',
        'lines-drw',
        'lines-fgf',
        'lines-bin',
        'lines-bxv',
        'lines-chy',
        'lines-ajr',
        'lines-dpt',
        'lines-efc',
        'lines-fzf',
        'lines-dno',
        'lines-cli',
        'lines-cii',
```

```
        'lines-bql',
        'lines-cts',
        'lines-cmc',
        'lines-ccg',
        'lines-ewi',
        'lines-dey',
        'lines-don',
        'lines-cdd',
        'lines-cxo',
        'lines-atq',
        'lines-ghn',
        'lines-cqp',
        'lines-chx',
        'lines-gfp',
        'lines-bvd',
        'lines-axw',
        'lines-cgt',
        'lines-gcv',
        'lines-cnj',
        'lines-gjg',
        'lines-dir',
        'lines-aan',
        'lines-gio',
        'lines-fqh',
        'lines-egs',
        'lines-cpg',
        'lines-alo',
        'lines-ahl',
        'lines-adq',
        'lines-fpo',
        'lines-aru',
        'lines-eqx',
        'lines-fop',
        'lines-fve',
        'lines-afz',
        'lines-cir',
        'lines-ewn',
        'lines-fix',
        'lines-epi',
        'lines-bma',
        'lines-bco',
        'lines-cpq',
        'lines-flb',
```

```
        'lines-aie',
        'lines-dxs',
        'lines-dvn',
        'lines-bak',
        'lines-fpz',
        'lines-eym',
        'lines-dag',
        'lines-abu',
        'lines-dfe',
        'lines-alc',
        'lines-gbl',
        'lines-aqw',
        'lines-cnw',
        'lines-axa',
        'lines-bxk',
        'lines-dnk',
        'lines-czu',
        'lines-edl',
        'lines-dxt',
        'lines-gel',
        'lines-ems',
        'lines-dll',
        'lines-fcg',
        'lines-fxs',
        'lines-brl',
        'lines-ddx',
        'lines-glp',
        'lines-ebm',
        'lines-bsa',
        'lines-ecs',
        'lines-bax',
        'lines-ctf',
        'lines-axr',
        'lines-auw',
        'lines-eao',
        'lines-ffw',
        'lines-glg',
        'lines-dcj',
        'lines-gaj',
        'lines-ehy',
        'lines-adj',
        'lines-aje',
        'lines-ckn',
        'lines-bfk',
```

```
        'lines-gke',
        'lines-gav',
        'lines-ghw',
        'lines-fwx',
        'lines-bdk',
        'lines-cfe',
        'lines-bzo',
        'lines-flm',
        'lines-fqt',
        'lines-dqb',
        'lines-fwm',
        'lines-ckr',
        'lines-glj',
        'lines-cqg',
        'lines-bet',
        'lines-axh',
        'lines-eja',
        'lines-djd',
        'lines-dou',
        'lines-egc',
        'lines-cct',
        'lines-efg',
        'lines-cme',
        'lines-avq',
        'lines-ens',
        'lines-gkz',
        'lines-cvd',
        'lines-bxl',
        'lines-bez',
        'lines-ekb',
        'lines-aau',
        'lines-fhc',
        'lines-dry',
        'lines-eii',
        'lines-acn',
        'lines-ggx',
        'lines-bei',
        'lines-brx',
        'lines-dhi',
        'lines-djs',
        'lines-bzf',
        'lines-dut',
        'lines-awk',
```

```
        'lines-feq',
        'lines-ges',
        'lines-bwi',
        'lines-cbu',
        'lines-bkw',
        'lines-ezv',
        'lines-fel',
        'lines-dca',
        'lines-exf',
        'lines-cjd',
        'lines-fxr',
        'lines-eld',
        'lines-euo',
        'lines-ggr',
        'lines-dcq',
        'lines-bls',
        'lines-bep',
        'lines-bvc',
        'lines-cdn',
        'lines-als',
        'lines-baw',
        'lines-aap',
        'lines-cwd',
        'lines-blp',
        'lines-csd',
        'lines-cie',
        'lines-bal',
        'lines-ckl',
        'lines-fdh',
        'lines-dof',
        'lines-csm',
        'lines-anj',
        'lines-ddh',
        'lines-gew',
        'lines-anl',
        'lines-dis',
        'lines-dgs',
        'lines-fnr',
        'lines-bsy',
        'lines-fbr',
        'lines-dox',
        'lines-fhr',
        'lines-enp',
        'lines-bcf',
```

```
        'lines-fsv',
        'lines-dow',
        'lines-ccl',
        'lines-fqa',
        'lines-fiw',
        'lines-bwj',
        'lines-eed',
        'lines-cyn',
        'lines-ctd',
        'lines-bmo',
        'lines-epv',
        'lines-fzn',
        'lines-cun',
        'lines-ajy',
        'lines-fbq',
        'lines-erw',
        'lines-bvg',
        'lines-bsm',
        'lines-fps',
        'lines-aai',
        'lines-flu',
        'lines-auc',
        'lines-fwp',
        'lines-gac',
        'lines-gca',
        'lines-byd',
        'lines-din',
        'lines-cec',
        'lines-dom',
        'lines-cxf',
        'lines-gfm',
        'lines-aed',
        'lines-bhq',
        'lines-esf',
        'lines-fzz',
        'lines-dlg',
        'lines-cbs',
        'lines-aup',
        'lines-gnk',
        'lines-bks',
        'lines-cwv',
        'lines-cax',
        'lines-ehr',
```

```
          'lines-frd',
          'lines-djp',
          'lines-bah',
          'lines-bcn',
          'lines-aql',
          'lines-efa',
          'lines-dld',
          'lines-fam',
          'lines-evk',
          'lines-gex',
          'lines-fvs',
          'lines-bui',
          'lines-cot',
          'lines-bub',
          'lines-bmf',
          'lines-akp',
          'lines-eqd',
          'lines-csl',
          'lines-daw',
          'lines-fmb',
          'lines-ank',
          'lines-fpb',
          'lines-btd',
          'lines-fmr',
          'lines-anp',
          'lines-btq',
          'lines-exs',
          'lines-dbb',
          'lines-bmx',
          'lines-ftb',
          'lines-avk',
          'lines-bhe',
          'lines-gkt',
          'lines-dqj',
          'lines-ech',
          'lines-cti',
          'lines-gjd',
          'lines-bnp',
          'lines-bej',
          'lines-buj',
          'lines-bij',
          'lines-gjn',
          'lines-dbe',
          'lines-abn',
```

```
        'lines-fnt',
        'lines-axv',
        'lines-dkl',
        'lines-fue',
        'lines-fcy',
        'lines-etv',
        'lines-diw',
        'lines-dip',
        'lines-bsg',
        'lines-gdb',
        'lines-glv',
        'lines-cxu',
        'lines-egr',
        'lines-akn',
        'lines-exo',
        'lines-eud',
        'lines-eca',
        'lines-all',
        'lines-bsx',
        'lines-cqb',
        'lines-dcn',
        'lines-edd',
        'lines-gdp',
        'lines-flx',
        'lines-asn',
        'lines-end',
        'lines-cfy',
        'lines-dgv',
        'lines-dkv',
        'lines-bmt',
        'lines-abt',
        'lines-gin',
        'lines-eyi',
        'lines-ety',
        'lines-bli',
        'lines-fce',
        'lines-cdk',
        'lines-ezz',
        'lines-cjf',
        'lines-bqf',
        'lines-exu',
        'lines-fxw',
        'lines-eht',
        'lines-dkf'
```

```
          'lines-dkf',
          'lines-fgp',
          'lines-alt',
          'lines-apd',
          'lines-dmf',
          'lines-dzy',
          'lines-dwf',
          'lines-agy',
          'lines-fsb',
          'lines-dic',
          'lines-azl',
          'lines-crl',
          'lines-dbj',
          'lines-cqc',
          'lines-cnt',
          'lines-eup',
          'lines-efe',
          'lines-dnq',
          'lines-dam',
          'lines-boc',
          'lines-git',
          'lines-axo',
          'lines-ceh',
          'lines-cwz',
          'lines-fbl',
          'lines-cte',
          'lines-ehz',
          'lines-ggf',
          'lines-azx',
          'lines-bke',
          'lines-bkf',
          'lines-cvg',
          'lines-fqu',
          'lines-dnx',
          'lines-biv',
          'lines-buo',
          'lines-asp',
          'lines-eyo',
          'lines-dyd',
          'lines-cdu',
          'lines-bct',
          'lines-ebj',
          'lines-cbd',
          'lines-fiz',
```

```
        'lines-dww',
        'lines-eyy',
        'lines-dar',
        'lines-fcc',
        'lines-dob',
        'lines-cbk',
        'lines-fds',
        'lines-des',
        'lines-dme',
        'lines-dkj',
        'lines-czw',
        'lines-bam',
        'lines-gii',
        'lines-afg',
        'lines-dcg',
        'lines-gdl',
        'lines-dxg',
        'lines-crp',
        'lines-bjb',
        'lines-bfp',
        'lines-dnu',
        'lines-cyi',
        'lines-ejh',
        'lines-fvf',
        'lines-fon',
        'lines-awc',
        'lines-dby',
        'lines-eas',
        'lines-dcs',
        'lines-gdi',
        'lines-dtq',
        'lines-fqp',
        'lines-feb',
        'lines-fwu',
        'lines-fcn',
        'lines-dnl',
        'lines-agd',
        'lines-ban',
        'lines-eoi',
        'lines-dxi',
        'lines-fhp',
        'lines-cee',
        'lines-dkh',
        'lines-ffq'
```

```
 'lines-ffq',
 'lines-baf',
 'lines-cnz',
 'lines-flf',
 'lines-dde',
 'lines-gdr',
 'lines-cvp',
 'lines-evu',
 'lines-ept',
 'lines-gco',
 'lines-fux',
 'lines-evo',
 'lines-bkq',
 'lines-bjd',
 'lines-ell',
 'lines-esq',
 'lines-bmd',
 'lines-bhk',
 'lines-dkc',
 'lines-gbn',
 'lines-cbm',
 'lines-ccj',
 'lines-fnh',
 'lines-ali',
 'lines-bhi',
 'lines-faj',
 'lines-bzr',
 'lines-eug',
 'lines-fuh',
 'lines-feh',
 'lines-fbe',
 'lines-dko',
 'lines-dcl',
 'lines-ehm',
 'lines-ffd',
 'lines-aja',
 'lines-bkb',
 'lines-ctt',
 'lines-fca',
 'lines-cxs',
 'lines-btr',
 'lines-dqk',
 'lines-cug',
 'lines-erv',
```

```
        'lines-epx',
        'lines-cgs',
        'lines-aqg',
        'lines-apk',
        'lines-bth',
        'lines-cxc',
        'lines-eky',
        'lines-dwq',
        'lines-azz',
        'lines-anv',
        'lines-fzu',
        'lines-fjh',
        'lines-axl',
        'lines-drg',
        'lines-dqu',
        'lines-gbc',
        'lines-fjm',
        'lines-clt',
        'lines-fei',
        'lines-ada',
        'lines-duv',
        'lines-byz',
        'lines-fkw',
        'lines-ghp',
        'lines-gji',
        'lines-ewk',
        'lines-acc',
        'lines-dcy',
        'lines-avv',
        'lines-aov',
        'lines-bhh',
        'lines-auq',
        'lines-ctn',
        'lines-dkd',
        'lines-eth',
        'lines-ayj',
        'lines-axy',
        'lines-exy',
        'lines-esa',
        'lines-exm',
        'lines-ghv',
        'lines-aox',
        'lines-azp',
        'lines-fko'
```

```
        'lines-rko',
        'lines-ebv',
        'lines-cgj',
        'lines-etf',
        'lines-dsb',
        'lines-cey',
        'lines-gbe',
        'lines-egx',
        'lines-crd',
        'lines-emh',
        'lines-arm',
        'lines-gfv',
        'lines-bpc',
        'lines-bvm',
        'lines-ezk',
        'lines-eeu',
        'lines-gag',
        'lines-aaj',
        'lines-duk',
        'lines-cyc',
        'lines-eyh',
        'lines-fzx',
        'lines-eql',
        'lines-afc',
        'lines-cus',
        'lines-axj',
        'lines-fyp',
        'lines-dtr',
        'lines-cih',
        'lines-fii',
        'lines-fbi',
        'lines-gei',
        'lines-aha',
        'lines-fhh',
        'lines-dod',
        'lines-dvk',
        'lines-ade',
        'lines-bsh',
        'lines-aao',
        'lines-blq',
        'lines-fte',
        'lines-bfj',
        'lines-epl',
        'lines-bia',
```

```
        'lines-geu',
        'lines-cer',
        'lines-bzu',
        'lines-gie',
        'lines-bog',
        'lines-ezj',
        'lines-akg',
        'lines-fvv',
        'lines-fpt',
        'lines-euu',
        'lines-atp',
        'lines-aqe',
        'lines-app',
        'lines-cij',
        'lines-bef',
        'lines-eeg',
        'lines-cbn',
        'lines-aro',
        'lines-esm',
        'lines-fuw',
        'lines-dbd',
        'lines-enw',
        'lines-fzr',
        'lines-bjy',
        'lines-evx',
        'lines-aar',
        'lines-acl',
        'lines-cgc',
        'lines-dlz',
        'lines-alf',
        'lines-acr',
        'lines-cmf',
        'lines-fxg',
        'lines-eom',
        'lines-bns',
        'lines-bju',
        'lines-das',
        'lines-ghf',
        'lines-efp',
        'lines-fmg',
        'lines-gby',
        'lines-erg',
        'lines-ami',
        'lines-dfu',
```

```
        'lines-arj',
        'lines-chd',
        'lines-dth',
        'lines-bis',
        'lines-bho',
        'lines-gan',
        'lines-adu',
        'lines-cdq',
        'lines-crt',
        'lines-day',
        'lines-frb',
        'lines-emr',
        'lines-dum',
        'lines-esh',
        'lines-ehq',
        'lines-djw',
        'lines-ead',
        'lines-dxn',
        'lines-cfu',
        'lines-cue',
        'lines-exq',
        'lines-elc',
        'lines-bwy',
        'lines-gch',
        'lines-adg',
        'lines-fjj',
        'lines-bve',
        'lines-adr',
        'lines-chm',
        'lines-ado',
        'lines-bip',
        'lines-bkt',
        'lines-fbf',
        'lines-dqd',
        'lines-dfm',
        'lines-dyx',
        'lines-cyl',
        'lines-fjz',
        'lines-ehp',
        'lines-csu',
        'lines-cwe',
        'lines-bdq',
        'lines-aqv',
```

```
        'lines-cmb',
        'lines-fow',
        'lines-evt',
        'lines-erf',
        'lines-djm',
        'lines-fen',
        'lines-anf',
        'lines-epc',
        'lines-bjh',
        'lines-fyw',
        'lines-gkn',
        'lines-bdo',
        'lines-bje',
        'lines-adf',
        'lines-evc',
        'lines-flj',
        'lines-dsk',
        'lines-edv',
        'lines-dlm',
        'lines-djt',
        'lines-cok',
        'lines-cgu',
        'lines-gmi',
        'lines-doy',
        'lines-bxa',
        'lines-fes',
        'lines-gdc',
        'lines-ald',
        'lines-egj',
        'lines-dvj',
        'lines-czk',
        'lines-eyv',
        'lines-amc',
        'lines-chc',
        'lines-epg',
        'lines-cfo',
        'lines-aur',
        'lines-ctk',
        'lines-etd',
        'lines-brn',
        'lines-eon',
        'lines-ezh',
        'lines-ayp',
        'lines-art',
```

```
        'lines-crb',
        'lines-bgo',
        'lines-dht',
        'lines-alb',
        'lines-dhq',
        'lines-fdg',
        'lines-fod',
        'lines-fgn',
        'lines-bts',
        'lines-ejr',
        'lines-dva',
        'lines-bdz',
        'lines-bzq',
        'lines-beh',
        'lines-awp',
        'lines-cgn',
        'lines-drr',
        'lines-fhx',
        'lines-dif',
        'lines-gjf',
        'lines-fuy',
        'lines-aak',
        'lines-alu',
        'lines-crx',
        'lines-cbx',
        'lines-axs',
        'lines-dta',
        'lines-fbx',
        'lines-cwc',
        'lines-bqq',
        'lines-azn',
        'lines-bmu',
        'lines-exc',
        'lines-biz',
        'lines-cke',
        'lines-ett',
        'lines-csr',
        'lines-ftj',
        'lines-euz',
        'lines-bel',
        'lines-gah',
        'lines-eqj',
        'lines-dzz',
```

```
        'lines-ftz',
        'lines-auk',
        'lines-fli',
        'lines-cta',
        'lines-fls',
        'lines-bvs',
        'lines-akl',
        'lines-dte',
        'lines-cip',
```

```
# Converts the unicode file to ascii
def unicode_to_ascii(s):
  return ''.join(c for c in unicodedata.normalize('NFD', s)
      if unicodedata.category(c) != 'Mn')


def preprocess_sentence(w):
  w = unicode_to_ascii(w.lower().strip())
  w = re.sub(r"([?.!,¿])", r" \1 ", w)
  w = re.sub(r'[" "]+', " ", w)
  w = re.sub(r"[^a-zA-Z?.!,¿]+", " ", w)
  w = w.strip()
  w = '<start> ' + w + ' <end>'
  return w
```

```
def line_cleaner(line):
  #line =""+str(line)
  line=line.decode("utf-8")
  #print(type(line))
  x=line.strip(" -.!?\n")
  return x
```

```
        'lines-fpn',
```

```
lines=[]
i=0
f= open('/tmp/data/lines-fli','rb')
for line in f:
  #print(line)
  line=line_cleaner(line)
  i=i+1
```

```
embed_dim=100
num_words=vocab_size
embedding_matrix=np.zeros((len(word_index)+1,embed_dim))
for word,i in word_index.items():
  embedding_vector=embeddings_index.get(word)
  if(embedding_vector is not None and i <num_words):
    embedding_matrix[i]=embedding_vector


from keras.layers import Embedding
embedding_layer=Embedding(len(word_index)+1,embed_dim,weights=[embedding_matrix],input_length=Tx,trainable=False)


#from faker import Faker
import random
from tqdm import tqdm
from babel.dates import format_date
#from nmt_utils import *
import matplotlib.pyplot as plt
%matplotlib inline



def softmax(x, axis=1):
    """Softmax activation function.
    # Arguments
        x : Tensor.
        axis: Integer, axis along which the softmax normalization is applied.
    # Returns
        Tensor, output of softmax transformation.
    # Raises
        ValueError: In case `dim(x) == 1`.
    """
    ndim = K.ndim(x)
    if ndim == 2:
        return K.softmax(x)
    elif ndim > 2:
        e = K.exp(x - K.max(x, axis=axis, keepdims=True))
        s = K.sum(e, axis=axis, keepdims=True)
        return e / s
```

```
        else:
            raise ValueError('Cannot apply softmax to a tensor that is 1D')


    # # Defined shared layers as global variables
    # repeator = RepeatVector(Tx)
    # concatenator = Concatenate(axis=-1)
    # densor1 = Dense(10, activation = "tanh")
    # densor2 = Dense(1, activation = "relu")
    # activator = Activation(softmax, name='attention_weights') # We are using a custom softmax(axis = 1) loaded in this notebook
    # dotor = Dot(axes = 1)


    # n_a = 128 # number of units for the pre-attention, bi-directional LSTM's hidden state 'a'
    # n_s = 128 # number of units for the post-attention LSTM's hidden state "s"

    # # Please note, this is the post attention LSTM cell.
    # # For the purposes of passing the automatic grader
    # # please do not modify this global variable.  This will be corrected once the automatic grader is also updated.
    # post_activation_LSTM_cell = LSTM(n_s, return_state = True) # post-attention LSTM
    # output_layer = Dense(10000, activation=softmax)


    # # GRADED FUNCTION: one_step_attention

    # def one_step_attention(a, s_prev):
    #     """
    #     Performs one step of attention: Outputs a context vector computed as a dot product of the attention weights
    #     "alphas" and the hidden states "a" of the Bi-LSTM.

    #     Arguments:
    #     a -- hidden state output of the Bi-LSTM, numpy-array of shape (m, Tx, 2*n_a)
    #     s_prev -- previous hidden state of the (post-attention) LSTM, numpy-array of shape (m, n_s)

    #     Returns:
    #     context -- context vector, input of the next (post-attention) LSTM cell
    #     """

    #     ### START CODE HERE ###
    #     # Use repeator to repeat s_prev to be of shape (m, Tx, n_s) so that you can concatenate it with all hidden states "a"
```

```
    #     s_prev = repeator(s_prev)
    #     # Use concatenator to concatenate a and s_prev on the last axis (≈ 1 line)
    #     # For grading purposes, please list 'a' first and 's_prev' second, in this order.
    #     concat = concatenator([a,s_prev])
    #     # Use densor1 to propagate concat through a small fully-connected neural network to compute the "intermediate energies
    #     e = densor1(concat)
    #     # Use densor2 to propagate e through a small fully-connected neural network to compute the "energies" variable energie
    #     energies = densor2(e)
    #     # Use "activator" on "energies" to compute the attention weights "alphas" (≈ 1 line)
    #     alphas = activator(energies)
    #     # Use dotor together with "alphas" and "a" to compute the context vector to be given to the next (post-attention) LSTM
    #     context = dotor([alphas,a])
    #     ### END CODE HERE ###

    #     return context


    # def model(Tx, Ty, n_a, n_s, human_vocab_size, machine_vocab_size):
    #     """
    #     Arguments:
    #     Tx -- length of the input sequence
    #     Ty -- length of the output sequence
    #     n_a -- hidden state size of the Bi-LSTM
    #     n_s -- hidden state size of the post-attention LSTM
    #     human_vocab_size -- size of the python dictionary "human_vocab"
    #     machine_vocab_size -- size of the python dictionary "machine_vocab"

    #     Returns:
    #     model -- Keras model instance
    #     """

    #     # Define the inputs of your model with a shape (Tx,)
    #     # Define s0 (initial hidden state) and c0 (initial cell state)
    #     # for the decoder LSTM with shape (n_s,)
    #     X = Input(shape=(Tx, human_vocab_size))
    #     s0 = Input(shape=(n_s,), name='s0')
    #     c0 = Input(shape=(n_s,), name='c0')
    #     s = s0
    #     c = c0
```

```
#     # Initialize empty list of outputs
#     outputs = []


#     ### START CODE HERE ###

#     # Step 1: Define your pre-attention Bi-LSTM. (≈ 1 line)
#     a = Bidirectional(LSTM(units=n_a, return_sequences=True))(X)

#     # Step 2: Iterate for Ty steps
#     for t in range(Ty):

#         # Step 2.A: Perform one step of the attention mechanism to get back the context vector at step t (≈ 1 line)
#         context = one_step_attention(a,s)

#         # Step 2.B: Apply the post-attention LSTM cell to the "context" vector.
#         # Don't forget to pass: initial_state = [hidden state, cell state] (≈ 1 line)
#         s, _, c = post_activation_LSTM_cell(context, initial_state= [s,c])

#         # Step 2.C: Apply Dense layer to the hidden state output of the post-attention LSTM (≈ 1 line)
#         out = output_layer(inputs=s)

#         # Step 2.D: Append "out" to the "outputs" list (≈ 1 line)
#         outputs.append(out)

#     # Step 3: Create model instance taking three inputs and returning the list of outputs. (≈ 1 line)
#     model = Model(inputs=[X,s0,c0], outputs=outputs)

#     ### END CODE HERE ###

#     return model


# model = model(Tx, Tx, n_a, n_s, 10000, 10000)
# model.summary()


# opt = Adam(lr = 0.005,beta_1 =  0.9,beta_2 =  0.999)
# model.compile(loss='categorical_crossentropy', optimizer='adam')
```

```
    def beam_search_decoder(prediction, BW = 3):

        output_sequences = [([], 0)]

        for sent in prediction:
            new_sequences = []
            sent = sent.reshape(20, )
            print(sent.shape)
            for old_seq, old_score in output_sequences:
                for char_index in range(len(sent)):
                    new_seq = old_seq + [char_index]
                    if(sent[char_index]==0):
                      continue
                    else:
                      new_score = old_score + math.log(sent[char_index])
                    new_sequences.append((new_seq, new_score))

            output_sequences = sorted(new_sequences, key = lambda val: val[1], reverse = True)
            output_sequences = output_sequences[:BW]

        return output_sequences
```

```
    if(i>=10000):
      break
    lines.append(line)
  print(i,lines)
```

```
10000 ['Okay, see you tonight', 'Yeah', 'Tonight', 'Hey, guys', 'Come on in', 'Good to see you', "Oh, I'm so excited to
```

```
train_data, validation_data = train_test_split(lines, test_size=.1, random_state=1234)
print(len(train_data))
```

```
8999
```

```
Tx=20
vocab_size=10000
padding='post'
trunc_type='post'
oov_tok='<OOV>'


#vecs
tokenizer=Tokenizer(num_words=vocab_size,oov_token=oov_tok)
tokenizer.fit_on_texts(lines)
word_index=tokenizer.word_index
sequences=tokenizer.texts_to_sequences(lines)
padded=pad_sequences(sequences,maxlen=Tx,padding=padding,truncating=trunc_type)
print(padded[:10])
```

```
      [[  40   71    2  107    0    0    0    0    0    0    0    0    0    0
           0    0    0    0    0    0]
       [  52    0    0    0    0    0    0    0    0    0    0    0    0    0
           0    0    0    0    0    0]
       [ 107    0    0    0    0    0    0    0    0    0    0    0    0    0
           0    0    0    0    0    0]
       [  53  123    0    0    0    0    0    0    0    0    0    0    0    0
           0    0    0    0    0    0]
       [  64   18   11    0    0    0    0    0    0    0    0    0    0    0
```

```
# #MODEL 2 tokenizer
# # Here I will use Tokenizer to extract the keyword vector as baseline
# # I will use train data to fit the Tokenizer, then use this Tokenizer to extract the validation data
# max_length = 100
# max_features = 50000
# token = Tokenizer(num_words=max_features)
# token.fit_on_texts(list(np.asarray(train_data.question_text)))
# xtrain = token.texts_to_sequences(np.asarray(train_data.question_text))
# xvalidate = token.texts_to_sequences(np.asarray(validation_data.question_text))
# xtest = token.texts_to_sequences(np.asarray(test.question_text))

# # Because Tokenizer will split the sentence, for some sentence are smaller,
# # so we have to pad the missing position
# xtrain = pad_sequences(xtrain, maxlen=max_length)
# xvalidate = pad_sequences(xvalidate, maxlen=max_length)
# xtest = pad_sequences(xtest, maxlen=max_length)

# ytrain = train_data.target
# yvaliate = validation_data.target


print(word_index)
print(len(word_index))
reverse_word_index={}
for word,ind in word_index.items():
  reverse_word_index[ind]=word
print(reverse_word_index)
```

```
{'<OOV>': 1, 'you': 2, 'the': 3, 'i': 4, 'to': 5, 'a': 6, 'and': 7, 'of': 8, 'it': 9, 'me': 10, 'in': 11, 'that': 12, '
3907
```

```
contexts=np.array([padded[i] for i in range(0,len(lines)-1)])
responses=np.array([padded[i] for i in range(1,len(lines))])
responses=np.expand_dims(responses,-1)
print(contexts[6])
print(responses[6])
print(contexts.shape,responses.shape)
```

```
[ 39  15  28 558   5  26   3 141   0   0   0   0   0   0   0   0   0   0
   0   0]
[[215]
 [  4]
 [ 21]
 [ 46]
 [  2]
 [  5]
 [369]
 [  0]
 [  0]
 [  0]
 [  0]
 [  0]
 [  0]
 [  0]
 [  0]
 [  0]
 [  0]
 [  0]
 [  0]
 [  0]]
(9998, 20) (9998, 20, 1)
```

```
!wget http://nlp.stanford.edu/data/glove.6B.zip
```

```
--2020-06-26 09:45:57--  http://nlp.stanford.edu/data/glove.6B.zip
Resolving nlp.stanford.edu (nlp.stanford.edu)... 171.64.67.140
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://nlp.stanford.edu/data/glove.6B.zip [following]
--2020-06-26 09:45:58--  https://nlp.stanford.edu/data/glove.6B.zip
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://downloads.cs.stanford.edu/nlp/data/glove.6B.zip [following]
--2020-06-26 09:45:59--  http://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
Resolving downloads.cs.stanford.edu (downloads.cs.stanford.edu)... 171.64.64.22
Connecting to downloads.cs.stanford.edu (downloads.cs.stanford.edu)|171.64.64.22|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 862182613 (822M) [application/zip]
Saving to: ‘glove.6B.zip’
```

```
from zipfile import ZipFile
file_name = "/content/glove.6B.zip"

with ZipFile(file_name, 'r') as zip:
  zip.extractall()
  print('Unzipped glove')
```

```
Unzipped glove
```

```
embeddings_index = {}
f = open( 'glove.6B.100d.txt')
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()

print('Number of word vectors is %s.' % len(embeddings_index))
```

```
Number of word vectors is 400000.
```

```
# model.compile(loss='categorical_crossentropy', optimizer='adam')


# s0=np.zeros((100,n_s))
# c0=np.zeros((100,n_s))
# model.fit([contexts], responses, epochs=20, batch_size=100)
#doesnot work due to a dimensionerror.


model=Sequential()
model.add(embedding_layer)
model.add(BatchNormalization())
model.add(LSTM(units=256,return_sequences=True))
model.add(TimeDistributed(Dense(vocab_size,activation='softmax')))


model.compile(loss='sparse_categorical_crossentropy',optimizer='adam')
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_1 (Embedding)      (None, 20, 100)           390800    
_________________________________________________________________
batch_normalization_1 (Batch (None, 20, 100)           400       
_________________________________________________________________
lstm_1 (LSTM)                (None, 20, 256)           365568    
_________________________________________________________________
time_distributed_1 (TimeDist (None, 20, 10000)         2570000   
=================================================================
Total params: 3,326,768
Trainable params: 2,935,768
Non-trainable params: 391,000
_________________________________________________________________
```

```
model.fit(contexts,responses,epochs=10,batch_size=10)
```

```
Epoch 1/10
9998/9998 [==============================] - 41s 4ms/step - loss: 1.0212
Epoch 2/10
9998/9998 [==============================] - 40s 4ms/step - loss: 0.9660
Epoch 3/10
9998/9998 [==============================] - 42s 4ms/step - loss: 0.9175
Epoch 4/10
9998/9998 [==============================] - 42s 4ms/step - loss: 0.8771
Epoch 5/10
9998/9998 [==============================] - 41s 4ms/step - loss: 0.8429
Epoch 6/10
9998/9998 [==============================] - 40s 4ms/step - loss: 0.8113
Epoch 7/10
9998/9998 [==============================] - 40s 4ms/step - loss: 0.7836
Epoch 8/10
9998/9998 [==============================] - 39s 4ms/step - loss: 0.7561
Epoch 9/10
9998/9998 [==============================] - 39s 4ms/step - loss: 0.7347
```

```
quest=["when are you coming home?"]
seq=tokenizer.texts_to_sequences(quest)
pad=pad_sequences(seq,maxlen=Tx,padding=padding,truncating=trunc_type)
```

```
pred=model.predict(pad)
pred=np.argmax(pred,axis=-1)
print(pred,pred.shape)
pred=pred.reshape(20,)
for i in pred:
  if i==0:
    continue
  else:
    print(reverse_word_index[i])
```

```
[[  4 807   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0]] (1, 20)
i
moment
```

```
import math
```