

```
{
  "cells": [
    {
      "cell_type": "markdown",
      "metadata": {},
      "source": [
        "# Creating a Sentiment Analysis Web App\n",
        "## Using PyTorch and SageMaker\n",
        "\n",
        "_Deep Learning Nanodegree Program | Deployment_\n",
        "\n",
        "---\n",
        "\n",
        "Now that we have a basic understanding of how SageMaker works we  

        will try to use it to construct a complete project from end to end.  

        Our goal will be to have a simple web page which a user can use to  

        enter a movie review. The web page will then send the review off to  

        our deployed model which will predict the sentiment of the entered  

        review.\n",
        "\n",
        "## Instructions\n",
        "\n",
        "Some template code has already been provided for you, and you  

        will need to implement additional functionality to successfully  

        complete this notebook. You will not need to modify the included code  

        beyond what is requested. Sections that begin with '**TODO**' in the  

        header indicate that you need to complete or implement some portion  

        within them. Instructions will be provided for each section and the  

        specifics of the implementation are marked in the code block with a `#  

        TODO: ...` comment. Please be sure to read the instructions carefully!  

        \n",
        "\n",
        "In addition to implementing code, there will be questions for you  

        to answer which relate to the task and your implementation. Each  

        section where you will answer a question is preceded by a  

        '**Question:**' header. Carefully read each question and provide your  

        answer below the '**Answer:**' header by editing the Markdown cell.  

        \n",
        "\n",
        "> **Note**: Code and Markdown cells can be executed using the  

        **Shift+Enter** keyboard shortcut. In addition, a cell can be edited  

        by typically clicking it (double-click for Markdown cells) or by  

        pressing **Enter** while it is highlighted.\n",
        "\n",
        "## General Outline\n",
        "\n",
        "Recall the general outline for SageMaker projects using a  

        notebook instance.\n",
        "\n",
        "1. Download or otherwise retrieve the data.\n",

```

```

    "2. Process / Prepare the data.\n",
    "3. Upload the processed data to S3.\n",
    "4. Train a chosen model.\n",
    "5. Test the trained model (typically using a batch transform
job).\n",
    "6. Deploy the trained model.\n",
    "7. Use the deployed model.\n",
    "\n",
    "For this project, you will be following the steps in the general
outline with some modifications. \n",
    "\n",
    "First, you will not be testing the model in its own step. You
will still be testing the model, however, you will do it by deploying
your model and then using the deployed model by sending the test data
to it. One of the reasons for doing this is so that you can make sure
that your deployed model is working correctly before moving forward.
\n",
    "\n",
    "In addition, you will deploy and use your trained model a second
time. In the second iteration you will customize the way that your
trained model is deployed by including some of your own code. In
addition, your newly deployed model will be used in the sentiment
analysis web app."
]
},
{
    "cell_type": "code",
    "execution_count": 54,
    "metadata": {},
    "outputs": [
        {
            "name": "stdout",
            "output_type": "stream",
            "text": [
                "Requirement already satisfied: sagemaker==1.72.0 in /home/ec2-
user/anaconda3/envs/pytorch_p36/lib/python3.6/site-packages
(1.72.0)\n",
                "Requirement already satisfied: boto3>=1.14.12 in /home/ec2-
user/anaconda3/envs/pytorch_p36/lib/python3.6/site-packages (from
sagemaker==1.72.0) (1.16.19)\n",
                "Requirement already satisfied: protobuf3-to-dict>=0.1.5 in /
home/ec2-user/anaconda3/envs/pytorch_p36/lib/python3.6/site-packages
(from sagemaker==1.72.0) (0.1.5)\n",
                "Requirement already satisfied: protobuf>=3.1 in /home/ec2-user/
anaconda3/envs/pytorch_p36/lib/python3.6/site-packages (from
sagemaker==1.72.0) (3.11.4)\n",
                "Requirement already satisfied: packaging>=20.0 in /home/ec2-
user/anaconda3/envs/pytorch_p36/lib/python3.6/site-packages (from
sagemaker==1.72.0) (20.1)\n",
                "Requirement already satisfied: smdebug-rulesconfig==0.1.4 in /

```

```

home/ec2-user/anaconda3/envs/pytorch_p36/lib/python3.6/site-packages
(from sagemaker==1.72.0) (0.1.4)\n",
    "Requirement already satisfied: scipy>=0.19.0 in /home/ec2-user/
anaconda3/envs/pytorch_p36/lib/python3.6/site-packages (from
sagemaker==1.72.0) (1.4.1)\n",
    "Requirement already satisfied: importlib-metadata>=1.4.0 in /
home/ec2-user/anaconda3/envs/pytorch_p36/lib/python3.6/site-packages
(from sagemaker==1.72.0) (2.0.0)\n",
    "Requirement already satisfied: numpy>=1.9.0 in /home/ec2-user/
anaconda3/envs/pytorch_p36/lib/python3.6/site-packages (from
sagemaker==1.72.0) (1.18.1)\n",
    "Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /home/
ec2-user/anaconda3/envs/pytorch_p36/lib/python3.6/site-packages (from
boto3>=1.14.12->sagemaker==1.72.0) (0.10.0)\n",
    "Requirement already satisfied: botocore<1.20.0,>=1.19.19 in /
home/ec2-user/anaconda3/envs/pytorch_p36/lib/python3.6/site-packages
(from boto3>=1.14.12->sagemaker==1.72.0) (1.19.19)\n",
    "Requirement already satisfied: s3transfer<0.4.0,>=0.3.0 in /
home/ec2-user/anaconda3/envs/pytorch_p36/lib/python3.6/site-packages
(from boto3>=1.14.12->sagemaker==1.72.0) (0.3.3)\n",
    "Requirement already satisfied: six in /home/ec2-user/anaconda3/
envs/pytorch_p36/lib/python3.6/site-packages (from protobuf3-to-
dict>=0.1.5->sagemaker==1.72.0) (1.14.0)\n",
    "Requirement already satisfied: setuptools in /home/ec2-user/
anaconda3/envs/pytorch_p36/lib/python3.6/site-packages (from
protobuf>=3.1->sagemaker==1.72.0) (45.2.0.post20200210)\n",
    "Requirement already satisfied: pyparsing>=2.0.2 in /home/ec2-
user/anaconda3/envs/pytorch_p36/lib/python3.6/site-packages (from
packaging>=20.0->sagemaker==1.72.0) (2.4.6)\n",
    "Requirement already satisfied: zipp>=0.5 in /home/ec2-user/
anaconda3/envs/pytorch_p36/lib/python3.6/site-packages (from
importlib-metadata>=1.4.0->sagemaker==1.72.0) (2.2.0)\n",
    "Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /
home/ec2-user/anaconda3/envs/pytorch_p36/lib/python3.6/site-packages
(from botocore<1.20.0,>=1.19.19->boto3>=1.14.12->sagemaker==1.72.0)
(2.8.1)\n",
    "Requirement already satisfied: urllib3<1.27,>=1.25.4 in /home/
ec2-user/anaconda3/envs/pytorch_p36/lib/python3.6/site-packages (from
botocore<1.20.0,>=1.19.19->boto3>=1.14.12->sagemaker==1.72.0)
(1.25.10)\n",
    "\u001b[33mWARNING: You are using pip version 20.0.2; however,
version 20.3.3 is available.\n",
    "You should consider upgrading via the '/home/ec2-user/
anaconda3/envs/pytorch_p36/bin/python -m pip install --upgrade pip'
command.\u001b[0m\n"
]
}
],
"source": [
    "# Make sure that we use SageMaker 1.x\n",

```

```

"!pip install sagemaker==1.72.0"
]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "## Step 1: Downloading the data\n",
    "\n",
    "As in the XGBoost in SageMaker notebook, we will be using the  

    [IMDb dataset](http://ai.stanford.edu/~amaas/data/sentiment/)\n",
    "\n",
    "> Maas, Andrew L., et al. [Learning Word Vectors for Sentiment  

    Analysis](http://ai.stanford.edu/~amaas/data/sentiment/). In  

    _Proceedings of the 49th Annual Meeting of the Association for  

    Computational Linguistics: Human Language Technologies_. Association  

    for Computational Linguistics, 2011."
  ]
},
{
  "cell_type": "code",
  "execution_count": 2,
  "metadata": {},
  "outputs": [
    {
      "name": "stdout",
      "output_type": "stream",
      "text": [
        "mkdir: cannot create directory '../data': File exists\n",
        "--2020-12-25 23:44:05-- http://ai.stanford.edu/~amaas/data/  

sentiment/aclImdb_v1.tar.gz\n",
        "Resolving ai.stanford.edu (ai.stanford.edu)... 171.64.68.10\n",
        "Connecting to ai.stanford.edu (ai.stanford.edu)|171.64.68.10|:  

80... connected.\n",
        "HTTP request sent, awaiting response... 200 OK\n",
        "Length: 84125825 (80M) [application/x-gzip]\n",
        "Saving to: '../data/aclImdb_v1.tar.gz'\n",
        "\n",
        "../data/aclImdb_v1. 100%[=====>] 80.23M 23.1MB/  

s in 4.8s \n",
        "\n",
        "2020-12-25 23:44:10 (16.8 MB/s) - '../data/aclImdb_v1.tar.gz'  

saved [84125825/84125825]\n",
        "\n"
      ]
    }
  ]
},
{
  "source": [
    "%mkdir ../data\n",
    "!wget -O ../data/aclImdb_v1.tar.gz http://ai.stanford.edu/~amaas/

```

```

data/sentiment/aclImdb_v1.tar.gz\n",
    "!tar -zxvf ../data/aclImdb_v1.tar.gz -C ../data"
],
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "## Step 2: Preparing and Processing the data\n",
        "\n",
        "Also, as in the XGBoost notebook, we will be doing some initial
data processing. The first few steps are the same as in the XGBoost
example. To begin with, we will read in each of the reviews and
combine them into a single input structure. Then, we will split the
dataset into a training set and a testing set."
    ]
},
{
    "cell_type": "code",
    "execution_count": 3,
    "metadata": {},
    "outputs": [],
    "source": [
        "import os\n",
        "import glob\n",
        "\n",
        "def read_imdb_data(data_dir='../data/aclImdb'):\n",
        "    data = {}\n",
        "    labels = {}\n",
        "    \n",
        "    for data_type in ['train', 'test']:\n",
        "        data[data_type] = {}\n",
        "        labels[data_type] = {}\n",
        "        \n",
        "        for sentiment in ['pos', 'neg']:\n",
        "            data[data_type][sentiment] = []\n",
        "            labels[data_type][sentiment] = []\n",
        "            \n",
        "            path = os.path.join(data_dir, data_type, sentiment,\n'*.txt')\n",
        "            files = glob.glob(path)\n",
        "            \n",
        "            for f in files:\n",
        "                with open(f) as review:\n",
        "                    data[data_type]\n[sentiment].append(review.read())\n",
        "                    # Here we represent a positive review by '1'\nand a negative review by '0'\n",
        "                    labels[data_type][sentiment].append(1 if\nsentiment == 'pos' else 0)

```

```

        "\n",
        "        assert len(data[data_type][sentiment]) ==
len(labels[data_type][sentiment]), \\n",
        "        \"{} / {} data size does not match labels
size\".format(data_type, sentiment)\n",
        "\n",
        "        return data, labels"
    ]
},
{
    "cell_type": "code",
    "execution_count": 4,
    "metadata": {},
    "outputs": [
        {
            "name": "stdout",
            "output_type": "stream",
            "text": [
                "IMDB reviews: train = 12500 pos / 12500 neg, test = 12500 pos /
12500 neg\n"
            ]
        }
    ],
    "source": [
        "data, labels = read_imdb_data()\n",
        "print(\"IMDB reviews: train = {} pos / {} neg, test = {} pos / {}
neg\".format(\n",
        "        len(data['train']['pos']), len(data['train']['neg']),
\n",
        "        len(data['test']['pos']), len(data['test']['neg'])))"
    ]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "Now that we've read the raw training and testing data from the
downloaded dataset, we will combine the positive and negative reviews
and shuffle the resulting records."
    ]
},
{
    "cell_type": "code",
    "execution_count": 5,
    "metadata": {},
    "outputs": [],
    "source": [
        "from sklearn.utils import shuffle\n",
        "\n",
        "def prepare_imdb_data(data, labels):\n",

```

```

    "    \"\"\"Prepare training and test sets from IMDb movie reviews.
\"\"\"\"\"n",
    "    \n",
    "    #Combine positive and negative reviews and labels\n",
    "    data_train = data['train']['pos'] + data['train']['neg']\n",
    "    data_test = data['test']['pos'] + data['test']['neg']\n",
    "    labels_train = labels['train']['pos'] + labels['train']
['neg']\n",
    "    labels_test = labels['test']['pos'] + labels['test']['neg']
\n",
    "    \n",
    "    #Shuffle reviews and corresponding labels within training and
test sets\n",
    "    data_train, labels_train = shuffle(data_train, labels_train)
\n",
    "    data_test, labels_test = shuffle(data_test, labels_test)\n",
    "    \n",
    "    # Return a unified training data, test data, training labels,
test labels\n",
    "    return data_train, data_test, labels_train, labels_test"
]
},
{
    "cell_type": "code",
    "execution_count": 6,
    "metadata": {},
    "outputs": [
        {
            "name": "stdout",
            "output_type": "stream",
            "text": [
                "IMDb reviews (combined): train = 25000, test = 25000\n"
            ]
        }
    ],
    "source": [
        "train_X, test_X, train_y, test_y = prepare_imdb_data(data,
labels)\n",
        "print(\"IMDb reviews (combined): train = {}, test = {}
\\.format(len(train_X), len(test_X))\"
    ]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "Now that we have our training and testing sets unified and
prepared, we should do a quick check and see an example of the data
our model will be trained on. This is generally a good idea as it
allows you to see how each of the further processing steps affects the

```

reviews and it also ensures that the data has been loaded correctly."

```
]
},
{
  "cell_type": "code",
  "execution_count": 7,
  "metadata": {},
  "outputs": [
    {
      "name": "stdout",
      "output_type": "stream",
      "text": [
        "I never expect a film adaptation to follow too closely to the
        novel (especially a beloved one, like Evening) but when I saw that the
        book's author, Susan Minot, was a screenplay writer and executive
        producer on the film, I thought that Evening would be a good
        adaptation.<br /><br />If you enjoyed the book, don't bother with this
        movie. It is so far afield of the book that the two hardly bear any
        resemblance to one another.<br /><br />Here, our characters are
        completely different: the bride is in love with Harris. Harris is the
        son of the housekeeper. Buddy is a drunk, in love with Ann and/or
        Harris. I don't think a single character made it from the book to the
        screen; oh it just gets worst with every passing moment.<br /><br /
        >And, really, didn't we learn from Bridges of Madison County that
        cutting from the story we are meant to be enthralled in, to scenes of
        our heroes' grown children having obnoxious and juvenile fights,
        simply does not work on film? This film is a disaster. Skip it.\n",
        "\n"
      ]
    }
  ],
  "source": [
    "print(train_X[100])\n",
    "print(train_y[100])"
  ]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "The first step in processing the reviews is to make sure that any
    html tags that appear should be removed. In addition we wish to
    tokenize our input, that way words such as *entertained* and
    *entertaining* are considered the same with regard to sentiment
    analysis."
  ]
},
{
  "cell_type": "code",
  "execution_count": 8,
```



```

"metadata": {},
"outputs": [],
"source": [
    "import nltk\n",
    "from nltk.corpus import stopwords\n",
    "from nltk.stem.porter import *\n",
    "\n",
    "import re\n",
    "from bs4 import BeautifulSoup\n",
    "\n",
    "def review_to_words(review):\n",
    "    nltk.download(\"stopwords\", quiet=True)\n",
    "    stemmer = PorterStemmer()\n",
    "    \n",
    "    text = BeautifulSoup(review, \"html.parser\").get_text() #
Remove HTML tags\n",
    "    text = re.sub(r\"[^\a-zA-Z0-9]\", \" \", text.lower()) #
Convert to lower case\n",
    "    words = text.split() # Split string into words\n",
    "    words = [w for w in words if w not in
stopwords.words(\"english\")] # Remove stopwords\n",
    "    words = [PorterStemmer().stem(w) for w in words] # stem\n",
    "    \n",
    "    return words"
]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "The `review_to_words` method defined above uses `BeautifulSoup`
to remove any html tags that appear and uses the `nltk` package to
tokenize the reviews. As a check to ensure we know how everything is
working, try applying `review_to_words` to one of the reviews in the
training set."
    ]
},
{
    "cell_type": "code",
    "execution_count": 9,
    "metadata": {},
    "outputs": [
        {
            "data": {
                "text/plain": [
                    "['never',\n",
                    " 'expect',\n",
                    " 'film',\n",
                    " 'adapt',\n",
                    " 'follow',\n"
                ]
            }
        ]
    ]
}

```

" 'close',\n",
" 'novel',\n",
" 'especi',\n",
" 'belov',\n",
" 'one',\n",
" 'like',\n",
" 'even',\n",
" 'saw',\n",
" 'book',\n",
" 'author',\n",
" 'susan',\n",
" 'minot',\n",
" 'screenplay',\n",
" 'writer',\n",
" 'execut',\n",
" 'produc',\n",
" 'film',\n",
" 'thought',\n",
" 'even',\n",
" 'would',\n",
" 'good',\n",
" 'adapt',\n",
" 'enjoy',\n",
" 'book',\n",
" 'bother',\n",
" 'movi',\n",
" 'far',\n",
" 'afield',\n",
" 'book',\n",
" 'two',\n",
" 'hardli',\n",
" 'bear',\n",
" 'resembl',\n",
" 'one',\n",
" 'anoth',\n",
" 'character',\n",
" 'complet',\n",
" 'differ',\n",
" 'bride',\n",
" 'love',\n",
" 'harri',\n",
" 'harri',\n",
" 'son',\n",
" 'housekeep',\n",
" 'buddi',\n",
" 'drunk',\n",
" 'love',\n",
" 'ann',\n",
" 'harri',\n",
" 'think',\n",

```

        " 'singl',\n",
        " 'charact',\n",
        " 'made',\n",
        " 'book',\n",
        " 'screen',\n",
        " 'oh',\n",
        " 'get',\n",
        " 'worst',\n",
        " 'everi',\n",
        " 'pass',\n",
        " 'moment',\n",
        " 'realli',\n",
        " 'learn',\n",
        " 'bridg',\n",
        " 'madison',\n",
        " 'counti',\n",
        " 'cut',\n",
        " 'stori',\n",
        " 'meant',\n",
        " 'enthral',\n",
        " 'scene',\n",
        " 'hero',\n",
        " 'grown',\n",
        " 'children',\n",
        " 'obnox',\n",
        " 'juvenil',\n",
        " 'fight',\n",
        " 'simpli',\n",
        " 'work',\n",
        " 'film',\n",
        " 'film',\n",
        " 'disast',\n",
        " 'skip']"
    ],
    "execution_count": 9,
    "metadata": {},
    "output_type": "execute_result"
}
],
"source": [
    "# TODO: Apply review_to_words to a review (train_X[100] or any
other review)\n",
    "review_to_words(train_X[100])"
]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [

```

Question: Above we mentioned that `review_to_words` method removes html formatting and allows us to tokenize the words found in a review, for example, converting `*entertained*` and `*entertaining*` into `*entertain*` so that they are treated as though they are the same word. What else, if anything, does this method do to the input?"

```
]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "**Answer:**\n",
    "It removes the stop words, removes the HTML tags, and makes
characters lowercase."
  ]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "The method below applies the review_to_words method to each of
the reviews in the training and testing datasets. In addition it
caches the results. This is because performing this processing step
can take a long time. This way if you are unable to complete the
notebook in the current session, you can come back without needing to
process the data a second time."
  ]
},
{
  "cell_type": "code",
  "execution_count": 10,
  "metadata": {},
  "outputs": [],
  "source": [
    "import pickle\n",
    "\n",
    "cache_dir = os.path.join("../cache", "sentiment_analysis") #
where to store cache files\n",
    "os.makedirs(cache_dir, exist_ok=True) # ensure cache directory
exists\n",
    "\n",
    "def preprocess_data(data_train, data_test, labels_train,
labels_test,\n",
    "                    cache_dir=cache_dir,
cache_file="preprocessed_data.pkl"):\n",
    "    \"\"\"Convert each review to words; read from cache if
available.\"\"\"\n",
    "    \n",
    "    # If cache_file is not None, try to read from it first\n",
    "    cache_data = None\n",

```

```

        "    if cache_file is not None:\n",
        "        try:\n",
        "            with open(os.path.join(cache_dir, cache_file),\n",
        "\"rb\") as f:\n",
        "                cache_data = pickle.load(f)\n",
        "                print(\"Read preprocessed data from cache file:\",\n",
        cache_file)\n",
        "            except:\n",
        "                pass # unable to read from cache, but that's\n",
        okay\n",
        "        \n",
        "        # If cache is missing, then do the heavy lifting\n",
        "        if cache_data is None:\n",
        "            # Preprocess training and test data to obtain words for\n",
        each review\n",
        "            #words_train = list(map(review_to_words, data_train))\n",
        "            #words_test = list(map(review_to_words, data_test))\n",
        "            words_train = [review_to_words(review) for review in\n",
        data_train]\n",
        "            words_test = [review_to_words(review) for review in\n",
        data_test]\n",
        "            \n",
        "            # Write to cache file for future runs\n",
        "            if cache_file is not None:\n",
        "                cache_data = dict(words_train=words_train,\n",
        words_test=words_test,\n",
        "                                labels_train=labels_train,\n",
        labels_test=labels_test)\n",
        "                with open(os.path.join(cache_dir, cache_file),\n",
        "\"wb\") as f:\n",
        "                    pickle.dump(cache_data, f)\n",
        "                    print(\"Wrote preprocessed data to cache file:\",\n",
        cache_file)\n",
        "            else:\n",
        "                # Unpack data loaded from cache file\n",
        "                words_train, words_test, labels_train, labels_test =\n",
        (cache_data['words_train'],\n",
        "         cache_data['words_test'],\n",
        cache_data['labels_train'], cache_data['labels_test'])\n",
        "            \n",
        "            return words_train, words_test, labels_train, labels_test"
    ]
},
{
    "cell_type": "code",
    "execution_count": 11,
    "metadata": {},
    "outputs": [
        {
            "name": "stdout",

```

```
"output_type": "stream",
"text": [
  "[0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0,
0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0,
1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1,
1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0,
1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1,
0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0,
1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0,
1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1,
1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1,
0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1,
0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1,
1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1,
0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1,
1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1,
0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1,
1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1,
0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1,
1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0,
0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0,
0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0,
1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1,
```

1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1,
1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1,
0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1,
0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1,
0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0,
0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0,
1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0,
1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1,
1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1,
1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1,
1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0,
0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,
0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1,
0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0,
0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1,
1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1,
1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1,
0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0,
1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0,
1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1,
0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0,
0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1,
1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1,
0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,
1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1,
1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1,
1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0,
1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1,

0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0,
1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1,
0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1,
1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1,
0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0,
0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0,
1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1,
1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1,
1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0,
0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1,
1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1,
0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1,
1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1,
1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1,
1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1,
0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1,
1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1,
1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1,
1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1,
0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0,
0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,
1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1,
1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1,
0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1,
1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1,
0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0,
0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1,
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0,
0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1,

0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1,
0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1,
0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0,
1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0,
1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1,
0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0,
1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1,
0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0,
1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0,
1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1,
0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0,
1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1,
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0,
1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0,
0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0,
0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,
1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1,
1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0,
0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0,
1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0,
0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1,
0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1,
1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0,
0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0,
0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1,
0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0,
1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0,
1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,

1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0,
0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1,
0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1,
0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1,
0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0,
1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0,
0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1,
1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0,
1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1,
1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0,
0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1,
0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0,
0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1,
0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1,
0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1,
0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1,
1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1,
0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1,
1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0,
0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1,
1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0,
1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1,
1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1,
0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,

[illegible]

0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0,
1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0,
1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0,
1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0,
1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0,
0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1,
1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0,
1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0,
0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0,
1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1,
0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0,
0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1,
0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1,
1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0,
1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1,
1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1,
1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1,
1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1,
1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1,
0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1,
0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0,
0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0,
1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1,
1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1,
1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0,
0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1,
1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1,

0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1,
1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1,
0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0,
1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1,
1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1,
1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0,
0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1,
1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0,
0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0,
1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0,
1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0,
1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,
1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,
1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0,
1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0,
1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0,
0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0,
0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0,
1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1,
0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0,
1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0,
0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0,
0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0,
1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1,
0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1,
0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1,
1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0,

0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0,
1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1,
0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0,
0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0,
1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0,
0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1,
0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0,
0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1,
0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1,
1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1,
1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1,
1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0,
0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1,
0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1,
0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0,
1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0,
1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0,
1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1,
1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1,
1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0,
1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1,
1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0,
1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,

0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0,
1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0,
1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0,
0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1,
0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0,
0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1,
0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1,
1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0,
0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0,
0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1,
0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0,
1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0,
1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1,
1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1,
0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1,
1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0,
1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0,
1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1,
0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1,
0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0,
1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1,
1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0,
1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1,
0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0,
1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1,
0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1,
1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0,
1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1,
0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0,

0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0,
0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0,
1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1,
0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1,
1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0,
1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0,
0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0,
1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0,
0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0,
0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0,
0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1,
1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0,
1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0,
0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1,
0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0,
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1,
1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1,
1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0,
0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0,
0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0,
0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0,
1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1,
1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1,
0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1,
0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0,
0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1,
0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0,
1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1,
0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0,
0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0,

1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1,
1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0,
0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,
0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1,
0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1,
1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0,
0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0,
1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0,
1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0,
0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0,
1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0,
1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0,
0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1,
1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1,
0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0,
1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1,
0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0,
0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1,
0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1,
0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1,
0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1,
1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1,
0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1,
1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1,
0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1,
1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0,

1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1,
1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1,
1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0,
1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0,
0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0,
1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0,
1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1,
1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0,
0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0,
1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1,
0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1,
0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1,
0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1,
1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1,
0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,
1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0,
0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,
0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1,
0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1,
0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1,
1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0,
1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0,
0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0,
1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0,
1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0,
0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0,

0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1,
0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1,
1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0,
0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0,
0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0,
1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1,
1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1,
0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0,
0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0,
0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1,
1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0,
0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0,
1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1,
0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1,
1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0,
1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0,
1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1,
1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1,
0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0,
1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1,
1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0,
1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1,
1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1,
1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1,
1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1,
1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0,
0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0,
1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1,
1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1,
1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1,
1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0,
0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0,
1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1,

0	1	1	0	0	0	0	1	0	1	1	0	1	0	1	0	0	0	1	1	0	0	1
0	1	0	0	1	0	0	0	1	0	0	1	0	1	1	1	0	1	1	1	1	1	0
1	0	1	1	1	0	0	0	0	1	0	1	1	0	1	1	0	0	1	0	1	0	0
0	0	1	1	0	0	0	0	1	0	1	1	1	1	1	1	0	1	1	0	1	0	0
0	0	1	1	0	1	1	0	1	1	0	1	0	0	1	1	0	1	1	0	1	0	1
0	0	0	0	1	0	0	0	0	1	0	1	0	0	1	1	0	0	0	1	1	0	0
0	0	1	0	1	0	0	0	0	0	1	0	1	1	1	1	1	0	1	1	0	1	1
1	1	0	1	0	0	1	1	1	1	1	0	0	0	1	0	1	0	1	1	1	0	1
1	1	1	1	1	1	0	1	0	0	0	1	0	1	1	1	0	1	1	1	1	1	0
0	1	1	0	0	1	1	0	1	0	1	0	0	1	0	1	0	0	1	1	0	0	1
1	0	1	1	0	1	1	1	0	0	1	0	1	0	1	1	1	0	1	1	0	1	1
0	1	0	0	0	1	0	1	1	0	1	1	0	0	1	0	0	0	1	0	1	1	0
1	0	1	0	1	0	1	0	1	0	1	1	0	0	0	0	0	1	1	1	0	1	0
1	0	0	1	1	1	1	0	1	0	1	1	0	0	1	1	1	1	1	0	0	1	1
1	1	0	1	1	0	1	1	0	1	1	1	0	0	0	1	1	0	1	0	0	1	1
1	0	1	0	1	1	0	0	1	0	1	0	1	0	0	0	1	1	0	0	1	0	1
1	0	1	1	0	1	0	0	1	0	1	0	1	0	0	1	1	0	1	1	0	1	0
1	0	0	1	0	0	1	1	0	1	0	1	0	0	1	1	0	0	1	0	0	0	1
1	1	1	0	0	1	1	0	0	0	1	1	1	1	1	1	0	0	1	1	1	0	1
0	0	1	1	1	1	1	1	1	0	1	0	0	1	0	0	0	1	1	1	0	0	0
1	0	0	1	1	0	0	0	1	1	1	1	1	1	0	1	1	0	1	0	0	1	1
1	1	1	0	0	1	0	0	0	1	1	1	1	1	1	0	1	1	0	0	0	1	0
0	0	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	1	1	0	0	1
1	0	0	1	0	0	1	0	0	1	1	0	0	1	0	1	0	0	0	0	1	0	0
1	0	0	1	0	0	0	0	1	1	0	1	0	0	1	1	0	1	1	0	1	0	1
0	0	0	0	1	0	1	0	1	1	1	1	0	0	0	0	0	1	0	1	0	1	1
1	1	1	1	0	1	1	0	1	1	1	1	0	0	0	0	1	1	1	1	1	0	1
1	0	0	0	1	1	1	0	0	0	0	0	0	0	1	0	1	0	1	1	0	1	0
0	1	0	0	1	0	1	1	1	1	0	1	0	0	0	0	0	0	1				

0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0,
0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0,
0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0,
0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1,
0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1,
0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0,
1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0,
0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1,
1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1,
0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0,
1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1,
0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1,
0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1,
0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1,
1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0,
0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0,
0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1,
0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0,
0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0,
0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0,
1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0,
1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1,
1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0,
1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0,
0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1,
1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1,
0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,

1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1,
0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0,
1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1,
0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0,
1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1,
0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1,
1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0,
0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1,
0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1,
0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0,
0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1,
0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0,
0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0,
0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1,
1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0,
1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0,
1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0,
1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1,
0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1,
0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0,

1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1,
0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0,
1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0,
0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1,
0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0,
1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1,
1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1,
1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1,
0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0,
0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0,
1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0,
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0,
0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0,
0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1,
0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0,
0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1,
0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0,
0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0,
0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0,
0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1,
1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1,
1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0,
0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1,
0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0,
1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1,
1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1,

0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0,
0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0,
0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1,
1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0,
0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0,
1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1,
1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1,
1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1,
0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1,
0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1,
1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1,
0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1,
0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1,
1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1,
1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1,
0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1,
1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0,
1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0,
0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0,
0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1,
0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1,
0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0,

0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1,
0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0,
0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0,
1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0,
0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1,
1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0,
1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1,
0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1,
1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1,
1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1,
1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1,
0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0,
0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0,
0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1,
0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1,
1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0,
1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1,
1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0,
1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0,
1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1,
1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1,
0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0,
1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1,
0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1,
0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1,
1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1,
1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1,

0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1,
1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1,
1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0,
0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1,
1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1,
0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0,
1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1,
0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0,
0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0,
0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1,
1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1,
1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0,
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0,
0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0,
1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0,
1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0,
0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0,
0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1,
1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0,
0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0,
0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0,
1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1,
1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0,
1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0,
1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1,
0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

```
1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1,
0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0,
0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1,
0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1,
1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0,
1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0,
1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1,
1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0,
0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1,
1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0,
1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1,
0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0,
1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0,
0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0,
0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1,
0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1,
1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,
0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1,
1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0,
0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0,
1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1,
1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0,
0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1,
```

```
1]\n"
]
},
{
  "name": "stderr",
  "output_type": "stream",
  "text": [
    "IOPub data rate exceeded.\n",
    "The notebook server will temporarily stop sending output\n",
    "to the client in order to avoid crashing it.\n",
    "To change this limit, set the config variable\n",
```

```

    "--NotebookApp.iopub_data_rate_limit`.\\n",
    "\\n",
    "Current values:\\n",
    "NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)\\n",
    "NotebookApp.rate_limit_window=3.0 (secs)\\n",
    "\\n"
  ]
}
],
"source": [
  "# Preprocess data\\n",
  "print(train_y)\\n",
  "train_X, test_X, train_y, test_y = preprocess_data(train_X,
test_X, train_y, test_y)\\n",
  "print(train_X)"
]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "## Transform the data\\n",
    "\\n",
    "In the XGBoost notebook we transformed the data from its word
representation to a bag-of-words feature representation. For the model
we are going to construct in this notebook we will construct a feature
representation which is very similar. To start, we will represent each
word as an integer. Of course, some of the words that appear in the
reviews occur very infrequently and so likely don't contain much
information for the purposes of sentiment analysis. The way we will
deal with this problem is that we will fix the size of our working
vocabulary and we will only include the words that appear most
frequently. We will then combine all of the infrequent words into a
single category and, in our case, we will label it as `1`.\\n",
    "\\n",
    "Since we will be using a recurrent neural network, it will be
convenient if the length of each review is the same. To do this, we
will fix a size for our reviews and then pad short reviews with the
category 'no word' (which we will label `0`) and truncate long
reviews."
  ]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "### (TODO) Create a word dictionary\\n",
    "\\n",
    "To begin with, we need to construct a way to map words that
appear in the reviews to integers. Here we fix the size of our

```

vocabulary (including the 'no word' and 'infrequent' categories) to be `5000` but you may wish to change this to see how it affects the model.\n",

"\n",
"> **TODO:** Complete the implementation for the `build_dict()` method below. Note that even though the vocab_size is set to `5000`, we only want to construct a mapping for the most frequently appearing `4998` words. This is because we want to reserve the special labels `0` for 'no word' and `1` for 'infrequent word'."

```
]
},
{
    "cell_type": "code",
    "execution_count": 80,
    "metadata": {},
    "outputs": [],
    "source": [
        "import numpy as np\n",
        "\n",
        "def build_dict(data, vocab_size = 5000, getTop5 = 0):\n",
        "    \"\"\"Construct and return a dictionary mapping each of the\n",
        "most frequently appearing words to a unique integer.\"\"\"\n",
        "    \n",
        "    # TODO: Determine how often each word appears in `data`. Note\n",
        "that `data` is a list of sentences and that a\n",
        "    # sentence is a list of words.\n",
        "    \n",
        "    word_count = {} # A dict storing the words that appear in the\n",
        "reviews along with how often they occur\n",
        "    \n",
        "    for each_review in data:\n",
        "        for each_word in each_review:\n",
        "            if each_word in word_count:\n",
        "                word_count[each_word] += 1\n",
        "            else:\n",
        "                word_count[each_word] = 1\n",
        "    \n",
        "    # TODO: Sort the words found in `data` so that\n",
        "sorted_words[0] is the most frequently appearing word and\n",
        "    # sorted_words[-1] is the least frequently appearing\n",
        "word.\n",
        "    \n",
        "    sorted_wordlist = sorted(word_count.items(), key=lambda x:\n",
        "x[1], reverse=True)\n",
        "    \n",
        "    sorted_words = [k for k,_ in sorted_wordlist]\n",
        "    \n",
        "    \n",
        "    if(getTop5==1):\n",
        "        return sorted_words
```

```

    "    \n",
    "    word_dict = {} # This is what we are building, a dictionary
that translates words into integers\n",
    "    for idx, word in enumerate(sorted_words[:vocab_size - 2]): #
The -2 is so that we save room for the 'no word'\n",
    "        word_dict[word] = idx + 2                                #
'infrequent' labels\n",
    "    return word_dict    \n",
    "\n",
    "
]
},
{
    "cell_type": "code",
    "execution_count": 81,
    "metadata": {},
    "outputs": [
        {
            "name": "stdout",
            "output_type": "stream",
            "text": [
                '{ 'movi': 2, 'film': 3, 'one': 4, 'like': 5, 'time': 6, 'good':
7, 'make': 8, 'character': 9, 'get': 10, 'see': 11, 'watch': 12,
'stori': 13, 'even': 14, 'would': 15, 'realli': 16, 'well': 17,
'scene': 18, 'look': 19, 'show': 20, 'much': 21, 'end': 22, 'peopl':
23, 'bad': 24, 'go': 25, 'great': 26, 'also': 27, 'first': 28, 'love':
29, 'think': 30, 'way': 31, 'act': 32, 'play': 33, 'made': 34,
'thing': 35, 'could': 36, 'know': 37, 'say': 38, 'seem': 39, 'work':
40, 'plot': 41, 'two': 42, 'actor': 43, 'year': 44, 'come': 45,
'mani': 46, 'seen': 47, 'take': 48, 'want': 49, 'life': 50, 'never':
51, 'littl': 52, 'best': 53, 'tri': 54, 'man': 55, 'ever': 56, 'give':
57, 'better': 58, 'still': 59, 'perform': 60, 'find': 61, 'feel': 62,
'part': 63, 'back': 64, 'use': 65, 'someth': 66, 'director': 67,
'actual': 68, 'interest': 69, 'lot': 70, 'real': 71, 'old': 72,
'cast': 73, 'though': 74, 'live': 75, 'star': 76, 'enjoy': 77, 'guy':
78, 'anoth': 79, 'new': 80, 'role': 81, 'noth': 82, '10': 83, 'funni':
84, 'music': 85, 'point': 86, 'start': 87, 'set': 88, 'girl': 89,
'origin': 90, 'day': 91, 'world': 92, 'everi': 93, 'believ': 94,
'turn': 95, 'quit': 96, 'direct': 97, 'us': 98, 'thought': 99, 'fact':
100, 'minut': 101, 'horror': 102, 'kill': 103, 'action': 104,
'comedi': 105, 'pretti': 106, 'young': 107, 'wonder': 108, 'happen':
109, 'around': 110, 'got': 111, 'effect': 112, 'right': 113, 'long':
114, 'howev': 115, 'big': 116, 'line': 117, 'famili': 118, 'enough':
119, 'seri': 120, 'may': 121, 'need': 122, 'fan': 123, 'bit': 124,
'script': 125, 'beauti': 126, 'person': 127, 'becom': 128, 'without':
129, 'must': 130, 'alway': 131, 'friend': 132, 'tell': 133, 'reason':
134, 'saw': 135, 'last': 136, 'final': 137, 'kid': 138, 'almost': 139,
'put': 140, 'least': 141, 'sure': 142, 'done': 143, 'whole': 144,
'place': 145, 'complet': 146, 'kind': 147, 'expect': 148, 'differ':
149, 'shot': 150, 'far': 151, 'mean': 152, 'anyth': 153, 'book': 154,

```

'laugh': 155, 'might': 156, 'name': 157, 'sinc': 158, 'begin': 159,
'2': 160, 'proabl': 161, 'woman': 162, 'help': 163, 'entertain': 164,
'let': 165, 'screen': 166, 'call': 167, 'tv': 168, 'moment': 169,
'away': 170, 'read': 171, 'yet': 172, 'rather': 173, 'worst': 174,
'run': 175, 'fun': 176, 'lead': 177, 'hard': 178, 'audienc': 179,
'idea': 180, 'anyon': 181, 'episod': 182, 'american': 183, 'found':
184, 'appear': 185, 'bore': 186, 'especi': 187, 'although': 188,
'hope': 189, 'cours': 190, 'keep': 191, 'anim': 192, 'job': 193,
'goe': 194, 'move': 195, 'sens': 196, 'dvd': 197, 'version': 198,
'war': 199, 'money': 200, 'someon': 201, 'mind': 202, 'mayb': 203,
'problem': 204, 'true': 205, 'hous': 206, 'everyth': 207, 'nice': 208,
'second': 209, 'rate': 210, 'three': 211, 'night': 212, 'face': 213,
'follow': 214, 'recommend': 215, 'main': 216, 'product': 217, 'worth':
218, 'leav': 219, 'human': 220, 'special': 221, 'excel': 222,
'togeth': 223, 'wast': 224, 'sound': 225, 'everyon': 226, 'john': 227,
'hand': 228, '1': 229, 'father': 230, 'later': 231, 'eye': 232,
'said': 233, 'view': 234, 'instead': 235, 'review': 236, 'boy': 237,
'high': 238, 'hour': 239, 'miss': 240, 'classic': 241, 'talk': 242,
'wife': 243, 'understand': 244, 'left': 245, 'care': 246, 'black':
247, 'death': 248, 'open': 249, 'murder': 250, 'write': 251, 'half':
252, 'head': 253, 'rememb': 254, 'chang': 255, 'viewer': 256, 'fight':
257, 'gener': 258, 'surpris': 259, 'short': 260, 'includ': 261, 'die':
262, 'fall': 263, 'less': 264, 'els': 265, 'entir': 266, 'piec': 267,
'involv': 268, 'pictur': 269, 'simpli': 270, 'power': 271, 'home':
272, 'top': 273, 'total': 274, 'usual': 275, 'budget': 276, 'attempt':
277, 'suppos': 278, 'releas': 279, 'hollywood': 280, 'terribl': 281,
'song': 282, 'men': 283, 'possibl': 284, 'featur': 285, 'portray':
286, 'disappoint': 287, '3': 288, 'poor': 289, 'coupl': 290, 'camera':
291, 'stupid': 292, 'dead': 293, 'wrong': 294, 'produc': 295, 'low':
296, 'video': 297, 'either': 298, 'aw': 299, 'definit': 300, 'except':
301, 'rest': 302, 'given': 303, 'absolut': 304, 'women': 305, 'lack':
306, 'word': 307, 'writer': 308, 'titl': 309, 'talent': 310, 'decid':
311, 'full': 312, 'perfect': 313, 'along': 314, 'style': 315, 'close':
316, 'truli': 317, 'school': 318, 'save': 319, 'emot': 320, 'sex':
321, 'age': 322, 'next': 323, 'bring': 324, 'mr': 325, 'case': 326,
'killer': 327, 'heart': 328, 'comment': 329, 'sort': 330, 'creat':
331, 'perhap': 332, 'came': 333, 'brother': 334, 'sever': 335, 'joke':
336, 'art': 337, 'dialogu': 338, 'game': 339, 'small': 340, 'base':
341, 'flick': 342, 'written': 343, 'sequenc': 344, 'meet': 345,
'earli': 346, 'often': 347, 'other': 348, 'mother': 349, 'develop':
350, 'humor': 351, 'actress': 352, 'consid': 353, 'dark': 354,
'guess': 355, 'amaz': 356, 'unfortun': 357, 'light': 358, 'lost': 359,
'exempl': 360, 'cinema': 361, 'drama': 362, 'ye': 363, 'white': 364,
'experi': 365, 'imagin': 366, 'mention': 367, 'stop': 368, 'natur':
369, 'forc': 370, 'manag': 371, 'felt': 372, 'present': 373, 'cut':
374, 'children': 375, 'fail': 376, 'son': 377, 'support': 378,
'qualiti': 379, 'car': 380, 'ask': 381, 'hit': 382, 'side': 383,
'voic': 384, 'extrem': 385, 'impress': 386, 'wors': 387, 'evil': 388,
'went': 389, 'stand': 390, 'certainli': 391, 'basic': 392, 'oh': 393,
'overall': 394, 'favorit': 395, 'horribl': 396, 'mysteri': 397,

'number': 398, 'type': 399, 'danc': 400, 'wait': 401, 'hero': 402, '5': 403, 'alreadi': 404, 'learn': 405, 'matter': 406, '4': 407, 'michael': 408, 'genr': 409, 'fine': 410, 'despit': 411, 'throughout': 412, 'walk': 413, 'success': 414, 'histori': 415, 'question': 416, 'zombi': 417, 'town': 418, 'relationship': 419, 'realiz': 420, 'past': 421, 'child': 422, 'daughter': 423, 'late': 424, 'b': 425, 'wish': 426, 'credit': 427, 'hate': 428, 'event': 429, 'theme': 430, 'touch': 431, 'citi': 432, 'today': 433, 'sometim': 434, 'behind': 435, 'god': 436, 'twist': 437, 'sit': 438, 'annoy': 439, 'deal': 440, 'stay': 441, 'abl': 442, 'rent': 443, 'pleas': 444, 'edit': 445, 'blood': 446, 'deserv': 447, 'anyway': 448, 'comic': 449, 'appar': 450, 'soon': 451, 'gave': 452, 'etc': 453, 'level': 454, 'slow': 455, 'chanc': 456, 'score': 457, 'bodi': 458, 'brilliant': 459, 'incred': 460, 'figur': 461, 'situat': 462, 'major': 463, 'self': 464, 'stuff': 465, 'decent': 466, 'element': 467, 'dream': 468, 'return': 469, 'obvious': 470, 'continu': 471, 'order': 472, 'pace': 473, 'ridicul': 474, 'happi': 475, 'add': 476, 'group': 477, 'highli': 478, 'thank': 479, 'ladi': 480, 'novel': 481, 'pain': 482, 'speak': 483, 'career': 484, 'shoot': 485, 'strang': 486, 'heard': 487, 'sad': 488, 'polic': 489, 'husband': 490, 'import': 491, 'break': 492, 'took': 493, 'cannot': 494, 'strong': 495, 'predict': 496, 'robert': 497, 'violenc': 498, 'hilari': 499, 'recent': 500, 'countri': 501, 'known': 502, 'particularli': 503, 'pick': 504, 'documentari': 505, 'season': 506, 'critic': 507, 'jame': 508, 'compar': 509, 'alon': 510, 'obviou': 511, 'told': 512, 'state': 513, 'rock': 514, 'visual': 515, 'theater': 516, 'exist': 517, 'offer': 518, 'opinion': 519, 'gore': 520, 'hold': 521, 'crap': 522, 'result': 523, 'realiti': 524, 'room': 525, 'hear': 526, 'clich': 527, 'effort': 528, 'thriller': 529, 'caus': 530, 'explain': 531, 'sequel': 532, 'serious': 533, 'king': 534, 'local': 535, 'ago': 536, 'none': 537, 'hell': 538, 'note': 539, 'allow': 540, 'david': 541, 'sister': 542, 'simpl': 543, 'femal': 544, 'deliv': 545, 'ok': 546, 'convinc': 547, 'class': 548, 'check': 549, 'suspens': 550, 'win': 551, 'buy': 552, 'oscar': 553, 'huge': 554, 'valu': 555, 'sexual': 556, 'scari': 557, 'cool': 558, 'similar': 559, 'excit': 560, 'provid': 561, 'exactli': 562, 'apart': 563, 'shown': 564, 'avoid': 565, 'seriou': 566, 'english': 567, 'whose': 568, 'taken': 569, 'cinematographi': 570, 'shock': 571, 'polit': 572, 'spoiler': 573, 'offic': 574, 'across': 575, 'middl': 576, 'pass': 577, 'street': 578, 'messag': 579, 'charm': 580, 'silli': 581, 'somewhat': 582, 'modern': 583, 'filmmak': 584, 'confus': 585, 'form': 586, 'tale': 587, 'singl': 588, 'jack': 589, 'mostli': 590, 'william': 591, 'attent': 592, 'carri': 593, 'sing': 594, 'subject': 595, 'five': 596, 'richard': 597, 'prove': 598, 'team': 599, 'stage': 600, 'unlik': 601, 'cop': 602, 'georg': 603, 'monster': 604, 'televis': 605, 'earth': 606, 'villain': 607, 'cover': 608, 'pay': 609, 'marri': 610, 'toward': 611, 'build': 612, 'parent': 613, 'pull': 614, 'due': 615, 'respect': 616, 'fill': 617, 'four': 618, 'dialog': 619, 'remind': 620, 'futur': 621, 'weak': 622, 'typic': 623, '7': 624, 'cheap': 625, 'intellig': 626, 'british': 627, 'atmospher': 628, '80': 629, 'clearli': 630, 'non': 631, 'paul': 632, 'dog': 633, 'artist': 634, '8': 635, 'knew':

636, 'fast': 637, 'crime': 638, 'easili': 639, 'escap': 640, 'doubt':
641, 'adult': 642, 'detail': 643, 'date': 644, 'member': 645, 'fire':
646, 'romant': 647, 'drive': 648, 'gun': 649, 'straight': 650,
'beyond': 651, 'fit': 652, 'attack': 653, 'imag': 654, 'upon': 655,
'posit': 656, 'whether': 657, 'fantast': 658, 'peter': 659, 'aspect':
660, 'captur': 661, 'appreci': 662, 'ten': 663, 'plan': 664, 'discov':
665, 'remain': 666, 'period': 667, 'near': 668, 'air': 669, 'realist':
670, 'mark': 671, 'red': 672, 'dull': 673, 'adapt': 674, 'within':
675, 'spend': 676, 'lose': 677, 'color': 678, 'materi': 679, 'chase':
680, 'mari': 681, 'storylin': 682, 'forget': 683, 'bunch': 684,
'clear': 685, 'lee': 686, 'victim': 687, 'nearli': 688, 'box': 689,
'york': 690, 'match': 691, 'inspir': 692, 'finish': 693, 'mess': 694,
'standard': 695, 'easi': 696, 'truth': 697, 'suffer': 698, 'busi':
699, 'bill': 700, 'space': 701, 'dramat': 702, 'western': 703, 'e':
704, 'list': 705, 'battl': 706, 'notic': 707, 'de': 708, 'french':
709, 'ad': 710, '9': 711, 'tom': 712, 'larg': 713, 'among': 714,
'eventu': 715, 'train': 716, 'accept': 717, 'agre': 718, 'soundtrack':
719, 'spirit': 720, 'third': 721, 'teenag': 722, 'adventur': 723,
'soldier': 724, 'drug': 725, 'sorri': 726, 'famou': 727, 'suggest':
728, 'normal': 729, 'cri': 730, 'babi': 731, 'ultim': 732, 'troubl':
733, 'contain': 734, 'certain': 735, 'cultur': 736, 'romanc': 737,
'rare': 738, 'lame': 739, 'somehow': 740, 'mix': 741, 'disney': 742,
'gone': 743, 'cartoon': 744, 'student': 745, 'reveal': 746, 'fear':
747, 'suck': 748, 'kept': 749, 'attract': 750, 'appeal': 751,
'premis': 752, 'design': 753, 'greatest': 754, 'secret': 755, 'shame':
756, 'throw': 757, 'copi': 758, 'scare': 759, 'wit': 760, 'admit':
761, 'america': 762, 'relat': 763, 'particular': 764, 'brought': 765,
'screenplay': 766, 'whatev': 767, 'pure': 768, '70': 769, 'averag':
770, 'harri': 771, 'master': 772, 'describ': 773, 'treat': 774,
'male': 775, '20': 776, 'issu': 777, 'fantasi': 778, 'warn': 779,
'inde': 780, 'forward': 781, 'background': 782, 'free': 783,
'project': 784, 'memor': 785, 'japanes': 786, 'poorli': 787, 'award':
788, 'locat': 789, 'potenti': 790, 'amus': 791, 'struggl': 792,
'magic': 793, 'weird': 794, 'societi': 795, 'okay': 796, 'doctor':
797, 'imdb': 798, 'accent': 799, 'hot': 800, 'water': 801, '30': 802,
'dr': 803, 'express': 804, 'alien': 805, 'odd': 806, 'choic': 807,
'crazi': 808, 'studio': 809, 'fiction': 810, 'becam': 811, 'control':
812, 'masterpiec': 813, 'fli': 814, 'difficult': 815, 'joe': 816,
'scream': 817, 'costum': 818, 'lover': 819, 'refer': 820, 'uniqu':
821, 'remak': 822, 'girlfriend': 823, 'vampir': 824, 'prison': 825,
'execut': 826, 'wear': 827, 'jump': 828, 'wood': 829, 'unless': 830,
'creepi': 831, 'cheesi': 832, 'superb': 833, 'otherwis': 834, 'parti':
835, 'roll': 836, 'ghost': 837, 'mad': 838, 'public': 839, 'depict':
840, 'badli': 841, 'week': 842, 'jane': 843, 'earlier': 844, 'moral':
845, 'dumb': 846, 'fi': 847, 'flaw': 848, 'grow': 849, 'deep': 850,
'sci': 851, 'maker': 852, 'cat': 853, 'older': 854, 'connect': 855,
'footag': 856, 'bother': 857, 'plenti': 858, 'outsid': 859, 'stick':
860, 'gay': 861, 'catch': 862, 'co': 863, 'plu': 864, 'popular': 865,
'equal': 866, 'social': 867, 'quickli': 868, 'disturb': 869,
'perfectli': 870, 'dress': 871, 'era': 872, '90': 873, 'mistak': 874,

'lie': 875, 'previou': 876, 'ride': 877, 'combin': 878, 'band': 879, 'concept': 880, 'surviv': 881, 'rich': 882, 'answer': 883, 'front': 884, 'sweet': 885, 'christma': 886, 'insid': 887, 'eat': 888, 'concern': 889, 'bare': 890, 'listen': 891, 'ben': 892, 'beat': 893, 'c': 894, 'term': 895, 'serv': 896, 'german': 897, 'meant': 898, 'la': 899, 'hardli': 900, 'stereotyp': 901, 'law': 902, 'innoc': 903, 'desper': 904, 'promis': 905, 'memori': 906, 'cute': 907, 'intent': 908, 'variou': 909, 'inform': 910, 'steal': 911, 'brain': 912, 'post': 913, 'tone': 914, 'island': 915, 'amount': 916, 'nuditi': 917, 'compani': 918, 'track': 919, 'claim': 920, 'store': 921, 'hair': 922, '50': 923, 'flat': 924, 'land': 925, 'univers': 926, 'kick': 927, 'danger': 928, 'fairli': 929, 'scott': 930, 'player': 931, 'plain': 932, 'crew': 933, 'step': 934, 'toni': 935, 'share': 936, 'tast': 937, 'centuri': 938, 'engag': 939, 'achiev': 940, 'travel': 941, 'cold': 942, 'record': 943, 'rip': 944, 'suit': 945, 'manner': 946, 'sadli': 947, 'spot': 948, 'tension': 949, 'wrote': 950, 'fascin': 951, 'intens': 952, 'familiar': 953, 'remark': 954, 'depth': 955, 'burn': 956, 'histor': 957, 'destroy': 958, 'sleep': 959, 'purpos': 960, 'languag': 961, 'ruin': 962, 'ignor': 963, 'delight': 964, 'unbeliev': 965, 'italian': 966, 'abil': 967, 'soul': 968, 'collect': 969, 'detect': 970, 'clever': 971, 'violent': 972, 'rape': 973, 'reach': 974, 'door': 975, 'scienc': 976, 'trash': 977, 'liter': 978, 'commun': 979, 'caught': 980, 'reveng': 981, 'creatur': 982, 'trip': 983, 'approach': 984, 'fashion': 985, 'intrigu': 986, 'skill': 987, 'introduc': 988, 'paint': 989, 'complex': 990, 'channel': 991, 'camp': 992, 'christian': 993, 'hole': 994, 'extra': 995, 'limit': 996, 'immedi': 997, 'mental': 998, 'ann': 999, 'million': 1000, '6': 1001, 'mere': 1002, 'comput': 1003, 'slightli': 1004, 'conclus': 1005, 'slasher': 1006, 'imposs': 1007, 'suddenli': 1008, 'teen': 1009, 'crimin': 1010, 'neither': 1011, 'physic': 1012, 'nation': 1013, 'spent': 1014, 'respons': 1015, 'planet': 1016, 'fake': 1017, 'receiv': 1018, 'sick': 1019, 'blue': 1020, 'bizarr': 1021, 'embarrass': 1022, 'indian': 1023, '15': 1024, 'ring': 1025, 'pop': 1026, 'drop': 1027, 'drag': 1028, 'haunt': 1029, 'suspect': 1030, 'pointless': 1031, 'search': 1032, 'edg': 1033, 'handl': 1034, 'biggest': 1035, 'common': 1036, 'faith': 1037, 'arriv': 1038, 'hurt': 1039, 'technic': 1040, 'angel': 1041, 'dad': 1042, 'genuin': 1043, 'awesom': 1044, 'f': 1045, 'solid': 1046, 'van': 1047, 'colleg': 1048, 'former': 1049, 'focu': 1050, 'count': 1051, 'heavi': 1052, 'tear': 1053, 'wall': 1054, 'rais': 1055, 'visit': 1056, 'laughabl': 1057, 'younger': 1058, 'sign': 1059, 'fair': 1060, 'excus': 1061, 'cult': 1062, 'motion': 1063, 'tough': 1064, 'key': 1065, 'super': 1066, 'desir': 1067, 'stun': 1068, 'addit': 1069, 'exploit': 1070, 'cloth': 1071, 'smith': 1072, 'tortur': 1073, 'race': 1074, 'davi': 1075, 'cross': 1076, 'author': 1077, 'jim': 1078, 'minor': 1079, 'compel': 1080, 'consist': 1081, 'focus': 1082, 'pathet': 1083, 'commit': 1084, 'chemistri': 1085, 'park': 1086, 'frank': 1087, 'obsess': 1088, 'tradit': 1089, 'grade': 1090, 'asid': 1091, '60': 1092, 'brutal': 1093, 'steve': 1094, 'somewher': 1095, 'depress': 1096, 'rule': 1097, 'opportun': 1098, 'explor': 1099, 'grant': 1100, 'u': 1101, 'honest':

1102, 'besid': 1103, 'anti': 1104, 'dub': 1105, 'trailer': 1106,
'intend': 1107, 'bar': 1108, 'regard': 1109, 'scientist': 1110,
'longer': 1111, 'west': 1112, 'decad': 1113, 'judg': 1114, 'silent':
1115, 'creativ': 1116, 'armi': 1117, 'wild': 1118, 'south': 1119, 'g':
1120, 'stewart': 1121, 'draw': 1122, 'road': 1123, 'govern': 1124,
'ex': 1125, 'boss': 1126, 'practic': 1127, 'club': 1128, 'motiv':
1129, 'festiv': 1130, 'surprisingli': 1131, 'gang': 1132, 'london':
1133, 'page': 1134, 'green': 1135, 'redeem': 1136, 'display': 1137,
'militari': 1138, 'idiot': 1139, 'aliv': 1140, 'machin': 1141,
'thrill': 1142, 'repeat': 1143, 'nobodi': 1144, 'folk': 1145, '100':
1146, 'yeah': 1147, '40': 1148, 'journey': 1149, 'garbag': 1150,
'smile': 1151, 'tire': 1152, 'ground': 1153, 'bought': 1154, 'mood':
1155, 'cost': 1156, 'sam': 1157, 'stone': 1158, 'mouth': 1159, 'noir':
1160, 'terrif': 1161, 'agent': 1162, 'requir': 1163, 'utterli': 1164,
'honestli': 1165, 'area': 1166, 'sexi': 1167, 'report': 1168, 'geniu':
1169, 'glad': 1170, 'enter': 1171, 'humour': 1172, 'investig': 1173,
'serial': 1174, 'passion': 1175, 'occasion': 1176, 'narr': 1177,
'climax': 1178, 'marriag': 1179, 'studi': 1180, 'industri': 1181,
'charli': 1182, 'center': 1183, 'ship': 1184, 'nowher': 1185, 'demon':
1186, 'bear': 1187, 'loos': 1188, 'hors': 1189, 'hang': 1190, 'wow':
1191, 'graphic': 1192, 'admir': 1193, 'giant': 1194, 'send': 1195,
'damn': 1196, 'loud': 1197, 'subtl': 1198, 'nake': 1199, 'rel': 1200,
'profession': 1201, 'blow': 1202, 'bottom': 1203, 'insult': 1204,
'batman': 1205, 'boyfriend': 1206, 'r': 1207, 'doubl': 1208, 'kelli':
1209, 'initi': 1210, 'frame': 1211, 'opera': 1212, 'gem': 1213,
'affect': 1214, 'church': 1215, 'challeng': 1216, 'drawn': 1217,
'cinemat': 1218, 'fulli': 1219, 'nightmar': 1220, 'l': 1221, 'seek':
1222, 'evid': 1223, 'j': 1224, 'essenti': 1225, 'conflict': 1226,
'arm': 1227, 'wind': 1228, 'henri': 1229, 'christoph': 1230, 'grace':
1231, 'narrat': 1232, 'witch': 1233, 'assum': 1234, 'push': 1235,
'hunt': 1236, 'chri': 1237, 'wise': 1238, 'month': 1239, 'repres':
1240, 'nomin': 1241, 'avail': 1242, 'hide': 1243, 'sceneri': 1244,
'affair': 1245, 'justic': 1246, 'smart': 1247, 'thu': 1248, 'bond':
1249, 'interview': 1250, 'outstand': 1251, 'flashback': 1252,
'constantli': 1253, 'presenc': 1254, 'satisfi': 1255, 'bed': 1256,
'central': 1257, 'sell': 1258, 'iron': 1259, 'content': 1260, 'gag':
1261, 'everybodi': 1262, 'hotel': 1263, 'slowli': 1264, 'hire': 1265,
'system': 1266, 'hey': 1267, 'thrown': 1268, 'individu': 1269, 'adam':
1270, 'charl': 1271, 'mediocr': 1272, 'jone': 1273, 'allen': 1274,
'lesson': 1275, 'ray': 1276, 'billi': 1277, 'photographi': 1278,
'cameo': 1279, 'fellow': 1280, 'pari': 1281, 'strike': 1282,
'independ': 1283, 'absurd': 1284, 'brief': 1285, 'rise': 1286, 'neg':
1287, 'impact': 1288, 'phone': 1289, 'born': 1290, 'ill': 1291,
'model': 1292, 'angl': 1293, 'spoil': 1294, 'fresh': 1295, 'abus':
1296, 'likabl': 1297, 'discuss': 1298, 'hill': 1299, 'ahead': 1300,
'sight': 1301, 'sent': 1302, 'photograph': 1303, 'logic': 1304,
'blame': 1305, 'shine': 1306, 'occur': 1307, 'bruce': 1308, 'mainli':
1309, 'commerci': 1310, 'skip': 1311, 'forev': 1312, 'teacher': 1313,
'held': 1314, 'segment': 1315, 'surround': 1316, 'zero': 1317,
'blond': 1318, 'summer': 1319, 'resembl': 1320, 'trap': 1321, 'satir':

1322, 'fool': 1323, 'queen': 1324, 'six': 1325, 'ball': 1326, 'sub':
1327, 'twice': 1328, 'tragedi': 1329, 'reaction': 1330, 'pack': 1331,
'bomb': 1332, 'protagonist': 1333, 'will': 1334, 'hospit': 1335,
'sport': 1336, 'mile': 1337, 'drink': 1338, 'jerri': 1339, 'mom':
1340, 'vote': 1341, 'trust': 1342, 'encount': 1343, 'plane': 1344,
'al': 1345, 'program': 1346, 'station': 1347, 'current': 1348,
'choos': 1349, 'martin': 1350, 'celebr': 1351, 'join': 1352, 'field':
1353, 'lord': 1354, 'round': 1355, 'tragic': 1356, 'favourit': 1357,
'jean': 1358, 'vision': 1359, 'robot': 1360, 'arthur': 1361, 'tie':
1362, 'fortun': 1363, 'random': 1364, 'roger': 1365, 'psycholog':
1366, 'dread': 1367, 'intern': 1368, 'nonsens': 1369, 'improv': 1370,
'prefer': 1371, 'epic': 1372, 'formula': 1373, 'highlight': 1374,
'pleasur': 1375, 'legend': 1376, 'dollar': 1377, 'tape': 1378, '11':
1379, 'porn': 1380, 'gorgeou': 1381, 'object': 1382, 'thin': 1383,
'wide': 1384, 'fox': 1385, 'buddi': 1386, 'ugli': 1387, 'influenc':
1388, 'nasti': 1389, 'ii': 1390, 'prepar': 1391, 'supposedli': 1392,
'progress': 1393, 'warm': 1394, 'reflect': 1395, 'youth': 1396,
'worthi': 1397, 'unusu': 1398, 'length': 1399, 'latter': 1400,
'crash': 1401, 'childhood': 1402, 'seven': 1403, 'superior': 1404,
'shop': 1405, 'remot': 1406, 'theatr': 1407, 'paid': 1408, 'disgust':
1409, 'funniest': 1410, 'pilot': 1411, 'fell': 1412, 'castl': 1413,
'trick': 1414, 'convers': 1415, 'rob': 1416, 'disast': 1417,
'gangster': 1418, 'establish': 1419, 'suicid': 1420, 'heaven': 1421,
'ident': 1422, 'disappear': 1423, 'mine': 1424, 'decis': 1425,
'heroin': 1426, 'tend': 1427, 'forgotten': 1428, 'singer': 1429,
'mask': 1430, 'partner': 1431, 'brian': 1432, 'desert': 1433, 'alan':
1434, 'recogn': 1435, 'sky': 1436, 'ms': 1437, 'p': 1438,
'thoroughli': 1439, 'stuck': 1440, 'accur': 1441, 'replac': 1442,
'market': 1443, 'commentari': 1444, 'uncl': 1445, 'clue': 1446,
'danni': 1447, 'eddi': 1448, 'seemingli': 1449, 'andi': 1450,
'jackson': 1451, 'devil': 1452, 'therefor': 1453, 'pair': 1454,
'refus': 1455, 'that': 1456, 'ed': 1457, 'fault': 1458, 'unit': 1459,
'accid': 1460, 'river': 1461, 'fate': 1462, 'tune': 1463, 'afraid':
1464, 'russian': 1465, 'clean': 1466, 'stephen': 1467, 'hidden': 1468,
'quick': 1469, 'instanc': 1470, 'irrit': 1471, 'captain': 1472,
'test': 1473, 'convey': 1474, 'readi': 1475, 'european': 1476,
'frustrat': 1477, 'insan': 1478, 'daniel': 1479, '1950': 1480, 'wed':
1481, 'rescu': 1482, 'food': 1483, 'chines': 1484, 'dirti': 1485,
'lock': 1486, 'angri': 1487, 'joy': 1488, 'steven': 1489, 'price':
1490, 'cage': 1491, 'bland': 1492, 'rang': 1493, 'anymor': 1494,
'wooden': 1495, 'jason': 1496, 'news': 1497, 'rush': 1498, 'n': 1499,
'led': 1500, 'martial': 1501, 'twenti': 1502, '12': 1503, 'worri':
1504, 'board': 1505, 'hunter': 1506, 'transform': 1507, 'symbol':
1508, 'cgi': 1509, 'onto': 1510, 'sentiment': 1511, 'piti': 1512, 'x':
1513, 'invent': 1514, 'johnni': 1515, 'attitud': 1516, 'process':
1517, 'explan': 1518, 'awar': 1519, 'owner': 1520, 'aim': 1521,
'floor': 1522, 'favor': 1523, 'necessari': 1524, 'target': 1525,
'energi': 1526, 'opposit': 1527, 'religi': 1528, 'blind': 1529,
'window': 1530, 'chick': 1531, 'insight': 1532, 'movement': 1533,
'possess': 1534, 'deepli': 1535, 'research': 1536, 'comparison': 1537,

'mountain': 1538, 'grand': 1539, 'whatsoev': 1540, 'comed': 1541,
'rain': 1542, 'began': 1543, 'mid': 1544, 'shadow': 1545, 'bank':
1546, 'princ': 1547, 'parodi': 1548, 'taylor': 1549, 'pre': 1550,
'weapon': 1551, 'friendship': 1552, 'credibl': 1553, 'teach': 1554,
'flesh': 1555, 'dougl'a': 1556, 'terror': 1557, 'bloodi': 1558, 'hint':
1559, 'protect': 1560, 'marvel': 1561, 'load': 1562, 'accord': 1563,
'anybodi': 1564, 'leader': 1565, 'drunk': 1566, 'watchabl': 1567,
'superman': 1568, 'freddi': 1569, 'brown': 1570, 'appropri': 1571,
'jeff': 1572, 'seat': 1573, 'tim': 1574, 'hitler': 1575, 'knock':
1576, 'villag': 1577, 'unknown': 1578, 'charg': 1579, 'keaton': 1580,
'enemi': 1581, 'media': 1582, 'england': 1583, 'unnecessari': 1584,
'empti': 1585, 'perspect': 1586, 'dare': 1587, 'craft': 1588, 'utter':
1589, 'buck': 1590, 'wave': 1591, 'strength': 1592, 'correct': 1593,
'ford': 1594, 'nativ': 1595, 'kiss': 1596, 'contrast': 1597, 'soap':
1598, 'anywher': 1599, 'knowledg': 1600, 'magnific': 1601, 'speed':
1602, 'nazi': 1603, 'chill': 1604, 'distract': 1605, '1980': 1606,
'breath': 1607, 'ice': 1608, 'fred': 1609, 'mission': 1610, 'moon':
1611, 'crowd': 1612, 'joan': 1613, 'jr': 1614, 'kate': 1615, 'soft':
1616, 'frighten': 1617, '000': 1618, 'nick': 1619, 'dan': 1620,
'hundr': 1621, 'dick': 1622, 'simon': 1623, 'somebodi': 1624, 'radio':
1625, 'dozen': 1626, 'loss': 1627, 'thousand': 1628, 'academi': 1629,
'andrew': 1630, 'shakespear': 1631, 'vehicl': 1632, 'quot': 1633,
'root': 1634, 'sum': 1635, 'account': 1636, '1970': 1637, 'convent':
1638, 'leg': 1639, 'behavior': 1640, 'gold': 1641, 'regular': 1642,
'compet': 1643, 'worker': 1644, 'pretenti': 1645, 'demand': 1646,
'notabl': 1647, 'privat': 1648, 'explos': 1649, 'candi': 1650,
'lynch': 1651, 'japan': 1652, 'stretch': 1653, 'interpret': 1654,
'constant': 1655, 'debut': 1656, 'tarzan': 1657, 'spi': 1658,
'revolv': 1659, 'sea': 1660, 'prais': 1661, 'translat': 1662,
'technolog': 1663, 'threaten': 1664, 'jesu': 1665, 'sat': 1666, 'ass':
1667, 'quiet': 1668, 'failur': 1669, 'franc': 1670, 'higher': 1671,
'met': 1672, 'punch': 1673, 'aid': 1674, 'kevin': 1675, 'toy': 1676,
'mike': 1677, 'vh': 1678, 'abandon': 1679, 'interact': 1680, 'bet':
1681, 'confront': 1682, 'command': 1683, 'separ': 1684, 'gotten':
1685, 'techniqu': 1686, 'recal': 1687, 'belong': 1688, 'site': 1689,
'servic': 1690, 'stunt': 1691, 'foot': 1692, 'freak': 1693, 'bug':
1694, 'cabl': 1695, 'fu': 1696, 'capabl': 1697, 'african': 1698,
'jimmi': 1699, 'bright': 1700, 'clark': 1701, 'fat': 1702, 'succeed':
1703, 'presid': 1704, 'boat': 1705, 'stock': 1706, 'gene': 1707,
'structur': 1708, 'spanish': 1709, 'paper': 1710, 'kidnap': 1711,
'belief': 1712, 'whilst': 1713, 'factor': 1714, 'realis': 1715,
'witti': 1716, 'complic': 1717, 'attend': 1718, 'bob': 1719, 'tree':
1720, 'educ': 1721, 'realism': 1722, 'assist': 1723, 'finest': 1724,
'santa': 1725, 'broken': 1726, 'smoke': 1727, 'up': 1728, 'observ':
1729, 'depart': 1730, 'v': 1731, 'determin': 1732, 'hat': 1733,
'routin': 1734, 'lewi': 1735, 'domin': 1736, 'fame': 1737, 'oper':
1738, 'rubbish': 1739, 'safe': 1740, 'kinda': 1741, 'lone': 1742,
'foreign': 1743, 'advanc': 1744, 'morgan': 1745, 'hook': 1746,
'numer': 1747, 'rank': 1748, 'washington': 1749, 'rose': 1750,
'werewolf': 1751, 'shallow': 1752, 'vs': 1753, 'shape': 1754, 'civil':

1755, 'gari': 1756, 'morn': 1757, 'accomplish': 1758, 'winner': 1759, 'kong': 1760, 'ordinari': 1761, 'grab': 1762, 'whenev': 1763, 'peac': 1764, 'virtual': 1765, 'luck': 1766, 'h': 1767, 'offens': 1768, 'unfunni': 1769, 'bigger': 1770, 'welcom': 1771, 'activ': 1772, 'patient': 1773, 'complain': 1774, 'contriv': 1775, 'pretend': 1776, 'con': 1777, 'dimension': 1778, 'trek': 1779, 'flash': 1780, 'eric': 1781, 'code': 1782, 'lesbian': 1783, 'dri': 1784, 'wake': 1785, 'cain': 1786, 'manipul': 1787, 'guard': 1788, 'statu': 1789, 'albert': 1790, 'corrupt': 1791, 'dancer': 1792, 'signific': 1793, 'awkward': 1794, 'context': 1795, 'gain': 1796, 'speech': 1797, 'sourc': 1798, '13': 1799, 'anthoni': 1800, 'sean': 1801, 'clip': 1802, 'psycho': 1803, 'corni': 1804, 'reli': 1805, 'curiou': 1806, 'theatric': 1807, 'advic': 1808, 'w': 1809, 'religion': 1810, 'priest': 1811, 'addict': 1812, 'flow': 1813, 'specif': 1814, 'jennif': 1815, 'skin': 1816, 'asian': 1817, 'secur': 1818, 'howard': 1819, 'core': 1820, 'promot': 1821, 'luke': 1822, 'organ': 1823, 'golden': 1824, 'comfort': 1825, 'cash': 1826, 'cheat': 1827, 'lucki': 1828, 'lower': 1829, 'dislik': 1830, 'associ': 1831, 'contribut': 1832, 'frequent': 1833, 'balanc': 1834, 'regret': 1835, 'wing': 1836, 'frankli': 1837, 'spell': 1838, 'devic': 1839, 'degre': 1840, 'sake': 1841, 'print': 1842, 'lake': 1843, 'forgiv': 1844, 'thoma': 1845, 'mass': 1846, 'betti': 1847, 'unexpect': 1848, 'gordon': 1849, 'crack': 1850, 'unfold': 1851, 'construct': 1852, 'categori': 1853, 'depend': 1854, 'amateur': 1855, 'grown': 1856, 'invit': 1857, 'anna': 1858, 'honor': 1859, 'intellectu': 1860, 'condit': 1861, 'walter': 1862, 'grew': 1863, 'matur': 1864, 'sudden': 1865, 'spectacular': 1866, 'sole': 1867, 'veteran': 1868, 'mirror': 1869, 'liner': 1870, 'gift': 1871, 'overli': 1872, 'grip': 1873, 'demonstr': 1874, 'experient': 1875, 'robin': 1876, 'meanwhil': 1877, 'freedom': 1878, 'card': 1879, 'crappi': 1880, 'sheriff': 1881, 'theori': 1882, 'section': 1883, 'subtitl': 1884, 'brilliantli': 1885, 'drew': 1886, 'unabl': 1887, 'colour': 1888, 'oliv': 1889, 'circumst': 1890, 'cook': 1891, 'parker': 1892, 'altern': 1893, 'matt': 1894, 'laughter': 1895, 'pile': 1896, 'path': 1897, 'sheer': 1898, 'defin': 1899, 'treatment': 1900, 'relief': 1901, 'hall': 1902, 'accident': 1903, 'wander': 1904, 'lawyer': 1905, 'sinatra': 1906, 'dragon': 1907, 'captiv': 1908, 'hank': 1909, 'moor': 1910, 'halloween': 1911, 'gratuit': 1912, 'broadway': 1913, 'kung': 1914, 'wound': 1915, 'k': 1916, 'cowboy': 1917, 'wayn': 1918, 'unintent': 1919, 'jacki': 1920, 'barbara': 1921, 'spoof': 1922, 'surreal': 1923, 'canadian': 1924, 'statement': 1925, 'winter': 1926, 'gonna': 1927, 'compos': 1928, 'treasur': 1929, 'fare': 1930, 'cheer': 1931, 'fish': 1932, 'sensit': 1933, 'emerg': 1934, 'unrealist': 1935, 'woodi': 1936, 'victor': 1937, 'neighbor': 1938, 'driven': 1939, 'sympathet': 1940, 'ran': 1941, 'menac': 1942, 'glass': 1943, 'authent': 1944, 'overlook': 1945, 'topic': 1946, 'expos': 1947, 'chief': 1948, 'handsom': 1949, 'ancient': 1950, 'gross': 1951, 'michel': 1952, 'pleasant': 1953, 'network': 1954, 'comedian': 1955, 'stranger': 1956, 'feet': 1957, 'built': 1958, 'nevertheless': 1959, 'contemporari': 1960, 'russel': 1961, 'cinderella': 1962, 'earn': 1963, 'letter': 1964, 'endless': 1965,

'underr': 1966, 'gori': 1967, 'consider': 1968, 'blockbust': 1969, 'miser': 1970, 'brook': 1971, 'switch': 1972, 'solv': 1973, 'edward': 1974, 'victoria': 1975, 'bullet': 1976, 'convict': 1977, 'virgin': 1978, 'joseph': 1979, 'chosen': 1980, 'cynic': 1981, '0': 1982, 'alex': 1983, 'scale': 1984, 'scenario': 1985, 'com': 1986, 'sword': 1987, 'outrag': 1988, 'gut': 1989, 'curs': 1990, 'screenwrit': 1991, 'wrap': 1992, 'juli': 1993, 'driver': 1994, 'monkey': 1995, 'uk': 1996, 'proper': 1997, 'substanc': 1998, 'indic': 1999, 'remov': 2000, 'bird': 2001, 'par': 2002, 'court': 2003, 'naiv': 2004, 'grave': 2005, 'inevit': 2006, 'nanci': 2007, 'rental': 2008, 'advertis': 2009, 'consequ': 2010, 'loser': 2011, 'roy': 2012, 'brave': 2013, 'le': 2014, 'slap': 2015, 'fatal': 2016, 'invis': 2017, 'bridg': 2018, 'germani': 2019, 'loui': 2020, 'provok': 2021, 'footbal': 2022, 'ador': 2023, 'anger': 2024, 'alcohol': 2025, 'anderson': 2026, 'chan': 2027, 'ryan': 2028, 'willi': 2029, 'stumbl': 2030, 'professor': 2031, 'sharp': 2032, 'australian': 2033, 'patrick': 2034, 'assassin': 2035, 'bat': 2036, '1930': 2037, 'amateurish': 2038, 'eight': 2039, 'saturday': 2040, 'trilog': 2041, 'deni': 2042, 'strongli': 2043, 'lousi': 2044, 'ape': 2045, 'cell': 2046, 'refresh': 2047, 'heck': 2048, 'liber': 2049, 'sin': 2050, 'vagu': 2051, 'resid': 2052, 'justifi': 2053, 'san': 2054, 'reput': 2055, 'defeat': 2056, 'indi': 2057, 'terrifi': 2058, 'creator': 2059, 'mini': 2060, 'sympathi': 2061, 'endur': 2062, 'task': 2063, 'tediou': 2064, 'prevent': 2065, 'tabl': 2066, 'expert': 2067, 'employ': 2068, 'trial': 2069, 'basebal': 2070, 'offend': 2071, 'rival': 2072, 'imit': 2073, 'che': 2074, 'max': 2075, 'fairi': 2076, 'dig': 2077, 'weekend': 2078, 'pitch': 2079, 'beach': 2080, 'europ': 2081, 'complaint': 2082, 'purchas': 2083, 'format': 2084, 'murphi': 2085, 'risk': 2086, 'harsh': 2087, 'reminisc': 2088, 'bite': 2089, 'glimps': 2090, 'hype': 2091, 'tini': 2092, 'nois': 2093, 'powel': 2094, 'titan': 2095, 'north': 2096, 'till': 2097, '14': 2098, 'strip': 2099, 'prime': 2100, 'asleep': 2101, 'fals': 2102, 'texa': 2103, 'revel': 2104, 'descript': 2105, 'africa': 2106, 'destruct': 2107, 'spin': 2108, 'arrest': 2109, 'inner': 2110, 'excess': 2111, 'sitcom': 2112, 'uninterest': 2113, 'semi': 2114, 'surfac': 2115, 'maintain': 2116, 'makeup': 2117, 'massiv': 2118, 'dinosaur': 2119, 'argu': 2120, 'twin': 2121, 'controversi': 2122, 'hitchcock': 2123, 'reject': 2124, 'ideal': 2125, 'stare': 2126, 'kim': 2127, 'expens': 2128, 'ludicr': 2129, 'insist': 2130, 'melodrama': 2131, 'supernatur': 2132, 'ala': 2133, 'forest': 2134, 'subplot': 2135, 'press': 2136, 'ga': 2137, 'atroci': 2138, 'columbo': 2139, 'host': 2140, 'erot': 2141, 'nail': 2142, 'presum': 2143, 'identifi': 2144, 'cant': 2145, 'notch': 2146, 'dude': 2147, 'plagu': 2148, 'forgett': 2149, 'guest': 2150, 'crude': 2151, 'closer': 2152, 'character': 2153, 'method': 2154, 'beast': 2155, 'princess': 2156, 'lion': 2157, 'landscap': 2158, 'border': 2159, 'ear': 2160, 'foster': 2161, 'urban': 2162, 'previous': 2163, 'damag': 2164, 'birth': 2165, 'pacino': 2166, 'aunt': 2167, 'storytel': 2168, 'bound': 2169, 'jungl': 2170, 'accus': 2171, 'thirti': 2172, 'chose': 2173, 'propaganda': 2174, 'doll': 2175, 'guid': 2176, 'nude': 2177, 'jess': 2178, 'emma': 2179, 'mainstream': 2180, 'whoever': 2181,

'mate': 2182, 'pet': 2183, '25': 2184, 'warrior': 2185, 'cooper': 2186, 'gritti': 2187, 'exact': 2188, 'merit': 2189, 'friday': 2190, 'poster': 2191, 'latest': 2192, 'upset': 2193, 'size': 2194, 'deadli': 2195, 'contact': 2196, 'contest': 2197, 'sun': 2198, 'rough': 2199, 'citizen': 2200, 'popul': 2201, 'buff': 2202, 'wilson': 2203, 'settl': 2204, 'blend': 2205, 'ton': 2206, 'corps': 2207, '1990': 2208, 'warner': 2209, 'rat': 2210, 'environ': 2211, 'pitt': 2212, 'alic': 2213, 'overcom': 2214, 'mgm': 2215, 'select': 2216, 'metal': 2217, 'widow': 2218, 'bu': 2219, 'particip': 2220, 'lift': 2221, 'ted': 2222, 'guilti': 2223, 'link': 2224, 'revolut': 2225, 'matrix': 2226, '1960': 2227, 'moron': 2228, 'afternoon': 2229, 'accompani': 2230, 'corner': 2231, 'prostitut': 2232, 'exagger': 2233, 'corpor': 2234, 'johnson': 2235, 'holm': 2236, 'hood': 2237, 'instal': 2238, 'multipl': 2239, 'leagu': 2240, 'doom': 2241, 'friendli': 2242, 'clair': 2243, 'sincer': 2244, 'defend': 2245, 'grim': 2246, 'campi': 2247, 'aka': 2248, 'junk': 2249, 'sunday': 2250, 'irish': 2251, 'string': 2252, 'examin': 2253, 'blah': 2254, 'advis': 2255, 'hip': 2256, 'lugosi': 2257, 'varieti': 2258, 'shut': 2259, 'tight': 2260, 'rachel': 2261, 'shake': 2262, 'icon': 2263, 'pro': 2264, 'confid': 2265, 'mexican': 2266, 'directli': 2267, 'goal': 2268, 'sullivan': 2269, 'medic': 2270, 'attach': 2271, 'jaw': 2272, 'denni': 2273, 'truck': 2274, 'courag': 2275, 'terrorist': 2276, 'sentenc': 2277, 'dean': 2278, 'prior': 2279, 'sarah': 2280, 'duke': 2281, 'breast': 2282, 'vietnam': 2283, 'legendari': 2284, 'bourn': 2285, 'entri': 2286, 'nose': 2287, 'donald': 2288, 'proceed': 2289, 'yell': 2290, 'behav': 2291, 'hong': 2292, 'un': 2293, 'split': 2294, 'buri': 2295, 'gather': 2296, 'unconvinc': 2297, 'jerk': 2298, 'everywher': 2299, 'forth': 2300, 'borrow': 2301, 'swim': 2302, 'confess': 2303, 'concentr': 2304, 'stolen': 2305, 'crush': 2306, 'lifetim': 2307, 'pan': 2308, 'spite': 2309, 'california': 2310, 'turkey': 2311, 'deliveri': 2312, 'julia': 2313, 'lip': 2314, 'proud': 2315, 'flight': 2316, 'quest': 2317, 'reward': 2318, 'offici': 2319, 'downright': 2320, 'freeman': 2321, 'hoffman': 2322, 'china': 2323, 'fade': 2324, 'sir': 2325, 'encourag': 2326, 'fabul': 2327, 'lazi': 2328, 'sink': 2329, 'jon': 2330, 'worthwhil': 2331, 'betray': 2332, 'jail': 2333, 'inept': 2334, 'notori': 2335, 'cousin': 2336, 'lisa': 2337, 'susan': 2338, 'relev': 2339, 'survivor': 2340, 'shower': 2341, 'bell': 2342, 'retard': 2343, 'teeth': 2344, 'bag': 2345, 'storm': 2346, 'imageri': 2347, 'branagh': 2348, 'tremend': 2349, 'trade': 2350, 'finger': 2351, 'mexico': 2352, 'bride': 2353, 'stab': 2354, 'summari': 2355, 'hugh': 2356, 'facial': 2357, 'shark': 2358, 'toler': 2359, 'quirki': 2360, 'alright': 2361, 'von': 2362, 'blown': 2363, 'ha': 2364, 'hyster': 2365, 'bitter': 2366, 'pose': 2367, 'larri': 2368, 'christ': 2369, 'bone': 2370, 'address': 2371, 'cruel': 2372, 'ned': 2373, 'afterward': 2374, 'scheme': 2375, 'ron': 2376, 'swear': 2377, 'traci': 2378, 'distinct': 2379, 'thumb': 2380, 'snake': 2381, 'pursu': 2382, 'feed': 2383, 'screw': 2384, 'tour': 2385, 'beg': 2386, 'mechan': 2387, 'obscur': 2388, 'chair': 2389, 'stomach': 2390, 'occas': 2391, 'raw': 2392, 'photo': 2393, 'necessarili': 2394, 'render': 2395, 'resist': 2396, 'heavili': 2397, 'southern': 2398,

'cabin': 2399, 'argument': 2400, 'gruesom': 2401, 'sidney': 2402, 'chain': 2403, 'holiday': 2404, 'hardi': 2405, 'understood': 2406, 'indulg': 2407, 'philip': 2408, 'satan': 2409, 'racist': 2410, 'india': 2411, 'forgot': 2412, 'outfit': 2413, 'belov': 2414, 'midnight': 2415, 'integr': 2416, 'stalk': 2417, 'obnoxi': 2418, 'lay': 2419, 'fourth': 2420, 'tongu': 2421, 'pregnant': 2422, 'inhabit': 2423, 'ticket': 2424, 'garden': 2425, 'restor': 2426, '17': 2427, 'magazin': 2428, 'slapstick': 2429, 'deeper': 2430, 'carol': 2431, 'brad': 2432, 'incid': 2433, 'shoe': 2434, 'devot': 2435, 'lincoln': 2436, 'disbelief': 2437, 'elizabeth': 2438, 'underground': 2439, 'lili': 2440, 'maria': 2441, 'benefit': 2442, 'guarante': 2443, 'divorc': 2444, 'anticip': 2445, 'sandler': 2446, 'capit': 2447, 'cring': 2448, 'creation': 2449, 'princip': 2450, 'mildli': 2451, 'slave': 2452, 'explod': 2453, 'greater': 2454, 'amazingli': 2455, 'bbc': 2456, 'lesli': 2457, 'extraordinari': 2458, 'introduct': 2459, 'halfway': 2460, 'funnier': 2461, 'overwhelm': 2462, 'wreck': 2463, 'advantag': 2464, 'extent': 2465, 'enhanc': 2466, 'punish': 2467, 'tap': 2468, 'transfer': 2469, 'text': 2470, 'lane': 2471, 'jessica': 2472, 'lo': 2473, 'preview': 2474, 'horrif': 2475, 'plant': 2476, 'east': 2477, 'dynam': 2478, 'error': 2479, 'deliber': 2480, 'miscast': 2481, 'homosexu': 2482, 'sophist': 2483, 'miller': 2484, 'appli': 2485, '2000': 2486, 'vincent': 2487, 'ensu': 2488, 'basi': 2489, 'vacat': 2490, 'elev': 2491, 'mansion': 2492, 'via': 2493, 'extend': 2494, 'bollywood': 2495, 'sleazi': 2496, 'uncomfort': 2497, 'reed': 2498, 'measur': 2499, 'spoken': 2500, 'steel': 2501, 'hippi': 2502, 'conceiv': 2503, 'goofi': 2504, 'mous': 2505, 'overact': 2506, 'blair': 2507, 'beer': 2508, 'melt': 2509, 'savag': 2510, 'dentist': 2511, 'cathol': 2512, 'daili': 2513, 'breathtak': 2514, 'assign': 2515, 'alter': 2516, 'fix': 2517, 'stanley': 2518, 'oppos': 2519, 'sacrific': 2520, 'subsequ': 2521, 'inspector': 2522, 'succe': 2523, 'burt': 2524, 'everyday': 2525, 'carpent': 2526, 'properli': 2527, 'nowaday': 2528, 'laura': 2529, 'circl': 2530, 'block': 2531, 'neck': 2532, 'massacr': 2533, 'fay': 2534, 'christi': 2535, 'mob': 2536, 'portrait': 2537, 'fallen': 2538, 'concert': 2539, 'pool': 2540, 'lesser': 2541, 'access': 2542, 'grey': 2543, 'seagal': 2544, 'jewish': 2545, 'relax': 2546, 'competit': 2547, 'sinist': 2548, 'chees': 2549, 'react': 2550, 'jake': 2551, 'usa': 2552, 'isol': 2553, 'ironi': 2554, 'nine': 2555, 'suitabl': 2556, 'stink': 2557, 'nonetheless': 2558, '2006': 2559, 'immens': 2560, 'appal': 2561, 'chop': 2562, 'creep': 2563, 'lyric': 2564, 'spiritu': 2565, 'rage': 2566, 'needless': 2567, 'navi': 2568, 'retir': 2569, 'sold': 2570, 'showcas': 2571, 'spring': 2572, 'reduc': 2573, 'user': 2574, 'shirt': 2575, 'nut': 2576, 'franchis': 2577, 'luci': 2578, 'adopt': 2579, 'zone': 2580, 'uninspir': 2581, 'nurs': 2582, 'per': 2583, 'digit': 2584, 'jay': 2585, 'bulli': 2586, 'bath': 2587, 'asham': 2588, 'stanwyck': 2589, '2001': 2590, 'sutherland': 2591, 'oddli': 2592, 'laid': 2593, 'illustr': 2594, 'amongst': 2595, 'upper': 2596, 'broadcast': 2597, '1940': 2598, 'baker': 2599, 'disguis': 2600, 'fulfil': 2601, 'throat': 2602, 'stylish': 2603, 'aspir': 2604, 'brando': 2605, 'wanna': 2606, 'impli': 2607, '18': 2608, 'pride':

2609, 'neighborhood': 2610, 'endear': 2611, 'nobl': 2612, 'thief':
2613, 'em': 2614, 'pound': 2615, 'wwii': 2616, 'cinematograph': 2617,
'prop': 2618, 'tens': 2619, 'shift': 2620, 'distribut': 2621, 'dawn':
2622, 'albeit': 2623, 'diseas': 2624, 'bo': 2625, 'dinner': 2626,
'16': 2627, 'shoulder': 2628, 'coher': 2629, 'bett': 2630, 'rochest':
2631, 'forti': 2632, 'function': 2633, 'poignant': 2634, 'knife':
2635, 'snow': 2636, 'surf': 2637, 'contract': 2638, 'matthau': 2639,
'wash': 2640, 'rebel': 2641, 'silenc': 2642, 'shout': 2643, 'cancel':
2644, 'derek': 2645, 'cannib': 2646, 'henc': 2647, 'height': 2648,
'silver': 2649, 'duti': 2650, 'proof': 2651, 'internet': 2652,
'horrend': 2653, 'heat': 2654, 'reunion': 2655, 'eeri': 2656,
'mindless': 2657, 'chuck': 2658, 'instinct': 2659, 'widmark': 2660,
'elvira': 2661, 'alik': 2662, 'spielberg': 2663, 'etern': 2664,
'torn': 2665, 'pie': 2666, 'musician': 2667, 'neat': 2668, 'absorb':
2669, 'premier': 2670, 'mill': 2671, 'glori': 2672, 'greatli': 2673,
'incoher': 2674, 'repetit': 2675, 'innov': 2676, 'elvi': 2677,
'itali': 2678, 'crisi': 2679, 'diamond': 2680, 'horrifi': 2681,
'redempt': 2682, 'bang': 2683, 'nelson': 2684, 'precis': 2685,
'infam': 2686, 'racism': 2687, 'announc': 2688, 'homag': 2689,
'britain': 2690, 'trite': 2691, 'wealthi': 2692, 'lovabl': 2693,
'fbi': 2694, 'blank': 2695, 'burton': 2696, 'dedic': 2697, 'hammer':
2698, 'helen': 2699, 'resolut': 2700, 'streisand': 2701, 'flop': 2702,
'ensembl': 2703, 'happili': 2704, 'chaplin': 2705, 'parallel': 2706,
'wilder': 2707, 'pat': 2708, 'plastic': 2709, 'triumph': 2710, 'st':
2711, 'broke': 2712, 'mar': 2713, 'factori': 2714, 'oil': 2715,
'conclud': 2716, 'disagre': 2717, 'carter': 2718, 'cube': 2719,
'rocket': 2720, 'weight': 2721, 'own': 2722, 'fighter': 2723, 'march':
2724, 'row': 2725, 'bush': 2726, 'climb': 2727, 'chuckl': 2728,
'vega': 2729, 'mst3k': 2730, 'kurt': 2731, 'boot': 2732, 'luca': 2733,
'lust': 2734, 'thug': 2735, 'enorm': 2736, 'wherea': 2737, 'dump':
2738, 'dane': 2739, 'spare': 2740, 'unforgett': 2741, 'sensibl': 2742,
'meaning': 2743, 'engin': 2744, 'stress': 2745, 'threat': 2746,
'brand': 2747, 'butt': 2748, 'difficulti': 2749, 'dear': 2750,
'fifti': 2751, 'adequ': 2752, 'caricatur': 2753, 'arnold': 2754,
'rap': 2755, 'karloff': 2756, 'bobbi': 2757, 'polish': 2758, 'fest':
2759, 'barri': 2760, 'arrog': 2761, 'ralph': 2762, 'swing': 2763,
'elabor': 2764, 'hamlet': 2765, 'secretari': 2766, 'journalist': 2767,
'flynn': 2768, 'ego': 2769, 'homeless': 2770, 'tool': 2771, 'resort':
2772, 'fanci': 2773, 'puppet': 2774, 'grate': 2775, 'simpson': 2776,
'induc': 2777, 'arrang': 2778, 'float': 2779, 'spike': 2780,
'conspiraci': 2781, 'unbear': 2782, 'cruis': 2783, 'tribut': 2784,
'muppet': 2785, 'pig': 2786, 'guilt': 2787, 'boll': 2788, 'exercis':
2789, 'phillip': 2790, 'choreograph': 2791, 'basement': 2792, 'item':
2793, '24': 2794, 'fianc': 2795, 'editor': 2796, 'puzzl': 2797,
'medium': 2798, 'toilet': 2799, 'file': 2800, 'tower': 2801, 'ham':
2802, 'layer': 2803, 'slip': 2804, 'ward': 2805, 'babe': 2806,
'scarecrow': 2807, 'korean': 2808, 'document': 2809, 'stan': 2810,
'glover': 2811, 'denzel': 2812, 'inexplic': 2813, 'assur': 2814,
'portion': 2815, 'transit': 2816, 'superfici': 2817, 'catherin': 2818,
'larger': 2819, 'philosoph': 2820, 'minim': 2821, 'slaughter': 2822,

'territori': 2823, 'librari': 2824, 'doc': 2825, 'orient': 2826, 'persona': 2827, 'spark': 2828, 'ban': 2829, 'owe': 2830, 'pg': 2831, 'sneak': 2832, 'jet': 2833, 'jeremi': 2834, 'financi': 2835, 'boredom': 2836, 'walken': 2837, 'wolf': 2838, 'dorothi': 2839, 'curti': 2840, 'shi': 2841, 'metaphor': 2842, 'eleph': 2843, 'cusack': 2844, 'multi': 2845, 'ambigu': 2846, 'backdrop': 2847, 'hudson': 2848, 'profound': 2849, 'whale': 2850, 'notion': 2851, '2005': 2852, 'elsewher': 2853, 'viru': 2854, 'stiff': 2855, 'birthday': 2856, 'ultra': 2857, 'union': 2858, 'hack': 2859, 'rave': 2860, 'gadget': 2861, 'implaus': 2862, 'lloyd': 2863, 'squar': 2864, 'newspap': 2865, 'deriv': 2866, 'afford': 2867, 'superhero': 2868, 'slight': 2869, 'pad': 2870, 'eva': 2871, '1st': 2872, 'disc': 2873, 'poison': 2874, 'bibl': 2875, 'reader': 2876, 'hawk': 2877, 'canada': 2878, 'distanc': 2879, 'eastwood': 2880, 'urg': 2881, 'spread': 2882, 'sadist': 2883, 'restaur': 2884, 'button': 2885, 'health': 2886, 'skit': 2887, 'essenc': 2888, 'charisma': 2889, 'drown': 2890, 'huh': 2891, 'montag': 2892, 'cure': 2893, 'heston': 2894, 'gradual': 2895, 'lab': 2896, 'invest': 2897, 'scoobi': 2898, 'dealt': 2899, 'maniac': 2900, 'godfath': 2901, 'companion': 2902, 'peak': 2903, 'muslim': 2904, 'estat': 2905, 'fetch': 2906, 'alli': 2907, 'gothic': 2908, 'kane': 2909, 'ritter': 2910, 'countless': 2911, 'cup': 2912, 'tea': 2913, 'subtleti': 2914, 'servant': 2915, 'miik': 2916, 'charismat': 2917, 'electr': 2918, 'iii': 2919, 'elect': 2920, 'briefli': 2921, 'salli': 2922, 'heroic': 2923, 'ingredi': 2924, 'wannab': 2925, 'resourc': 2926, 'tender': 2927, 'bud': 2928, 'nuanc': 2929, 'grandmoth': 2930, 'toss': 2931, 'admittedli': 2932, 'neil': 2933, 'reel': 2934, 'cole': 2935, 'dawson': 2936, 'poverti': 2937, 'carrey': 2938, 'shall': 2939, 'mafia': 2940, 'stood': 2941, 'pauli': 2942, 'label': 2943, 'mild': 2944, 'stronger': 2945, 'pit': 2946, 'kubrick': 2947, 'reev': 2948, 'gate': 2949, 'punk': 2950, 'cox': 2951, 'assault': 2952, 'bakshi': 2953, 'fond': 2954, 'cardboard': 2955, 'useless': 2956, 'updat': 2957, 'tag': 2958, 'easier': 2959, 'smash': 2960, 'smooth': 2961, 'ian': 2962, 'astair': 2963, 'burst': 2964, 'outcom': 2965, 'terri': 2966, '2002': 2967, 'vari': 2968, 'divers': 2969, 'vulner': 2970, 'increasingli': 2971, 'resolv': 2972, 'coincid': 2973, 'samurai': 2974, 'melodramat': 2975, 'qualifi': 2976, 'exchang': 2977, 'sketch': 2978, 'rex': 2979, 'fist': 2980, 'scratch': 2981, 'brilliant': 2982, 'tame': 2983, 'reynold': 2984, 'conveni': 2985, 'insert': 2986, 'luckili': 2987, 'be': 2988, 'blast': 2989, 'templ': 2990, 'suspend': 2991, 'meat': 2992, 'fisher': 2993, 'strictli': 2994, 'matthew': 2995, 'coach': 2996, 'farm': 2997, 'seventi': 2998, 'soprano': 2999, 'walker': 3000, 'ambiti': 3001, 'gotta': 3002, 'hamilton': 3003, 'nuclear': 3004, 'jami': 3005, 'pin': 3006, 'recreat': 3007, 'eccentr': 3008, 'discoveri': 3009, 'empir': 3010, 'worthless': 3011, 'cave': 3012, 'ninja': 3013, 'butcher': 3014, 'spooki': 3015, 'clock': 3016, 'kudo': 3017, 'instantli': 3018, 'monk': 3019, 'timeless': 3020, 'convolut': 3021, 'joey': 3022, 'grasp': 3023, 'brosnan': 3024, 'revers': 3025, 'closet': 3026, 'struck': 3027, 'communist': 3028, 'sloppi': 3029, 'importantli': 3030, 'mitchel': 3031, 'fifteen': 3032, 'gray': 3033, 'declar': 3034, 'miracl': 3035, 'pal': 3036, 'wipe':

3037, 'inconsist': 3038, 'eighti': 3039, 'partli': 3040, 'clown':
3041, 'seller': 3042, 'norman': 3043, 'sidekick': 3044, 'evok': 3045,
'cliff': 3046, 'bleak': 3047, 'selfish': 3048, 'debat': 3049, 'piano':
3050, 'websit': 3051, 'farc': 3052, 'australia': 3053, 'cheek': 3054,
'chew': 3055, 'destin': 3056, 'superbl': 3057, 'aforement': 3058,
'lifestyl': 3059, 'enthusiast': 3060, 'psychiatrist': 3061,
'flawless': 3062, 'ho': 3063, '45': 3064, 'stoog': 3065, 'seed': 3066,
'wrestl': 3067, 'akshay': 3068, 'regardless': 3069, 'soviet': 3070,
'director': 3071, 'splatter': 3072, 'pressur': 3073, 'emili': 3074,
'dash': 3075, 'kitchen': 3076, 'slice': 3077, 'dire': 3078, 'drivel':
3079, 'incompet': 3080, 'bash': 3081, 'anni': 3082, 'wick': 3083,
'abc': 3084, 'ken': 3085, 'cameron': 3086, 'doo': 3087, 'lou': 3088,
'helicopt': 3089, 'duo': 3090, 'flower': 3091, 'prize': 3092,
'increas': 3093, 'pleasantli': 3094, 'dave': 3095, 'beaten': 3096,
'chapter': 3097, 'recov': 3098, 'judi': 3099, 'boil': 3100,
'artifici': 3101, 'glow': 3102, 'cia': 3103, 'seduc': 3104, 'curios':
3105, 'suppli': 3106, 'blob': 3107, 'distant': 3108, 'jar': 3109,
'mann': 3110, 'cagney': 3111, 'goldberg': 3112, 'eleg': 3113, 'ellen':
3114, 'craven': 3115, 'glenn': 3116, 'panic': 3117, 'psychot': 3118,
'combat': 3119, 'craig': 3120, 'ranger': 3121, 'favour': 3122,
'splendid': 3123, 'laurel': 3124, 'francisco': 3125, 'turner': 3126,
'web': 3127, 'hop': 3128, 'perri': 3129, 'drunken': 3130, 'modesti':
3131, 'shortli': 3132, 'graduat': 3133, 'rid': 3134, 'slightest':
3135, 'gentl': 3136, 'ruth': 3137, '20th': 3138, 'hatr': 3139, 'falk':
3140, 'min': 3141, 'gandhi': 3142, 'philosophi': 3143, 'plausibl':
3144, 'wizard': 3145, 'flip': 3146, 'fx': 3147, 'alexand': 3148,
'greek': 3149, 'manhattan': 3150, 'futurist': 3151, 'ocean': 3152,
'knight': 3153, 'we': 3154, 'unpleas': 3155, 'tall': 3156, 'jealou':
3157, 'preciou': 3158, 'legal': 3159, 'dracula': 3160, 'holi': 3161,
'harm': 3162, 'fund': 3163, 'felix': 3164, 'lend': 3165, 'scientif':
3166, 'bless': 3167, 'ami': 3168, 'reviv': 3169, 'mock': 3170,
'forbidden': 3171, 'nod': 3172, 'overdon': 3173, 'digniti': 3174,
'giallo': 3175, 'explicit': 3176, 'tank': 3177, 'thread': 3178,
'childish': 3179, 'unwatch': 3180, 'eve': 3181, 'elderli': 3182,
'repeatedli': 3183, 'awaken': 3184, 'torment': 3185, 'yesterday':
3186, 'nerv': 3187, 'broad': 3188, 'awe': 3189, '2004': 3190, 'fever':
3191, 'pirat': 3192, 'verhoeven': 3193, 'mel': 3194, 'thick': 3195,
'margaret': 3196, '99': 3197, 'ambit': 3198, 'lean': 3199, 'stiller':
3200, 'politician': 3201, 'roman': 3202, 'romero': 3203, 'automat':
3204, 'custom': 3205, 'timothi': 3206, 'rivet': 3207, 'acclaim': 3208,
'eas': 3209, 'bin': 3210, 'ah': 3211, 'publish': 3212, 'kay': 3213,
'griffith': 3214, 'launch': 3215, 'absenc': 3216, 'uniform': 3217,
'royal': 3218, 'stinker': 3219, 'phrase': 3220, 'darker': 3221,
'purpl': 3222, 'gabriel': 3223, 'foul': 3224, 'antic': 3225,
'homicid': 3226, 'warren': 3227, 'crook': 3228, 'bathroom': 3229,
'pulp': 3230, 'termin': 3231, 'pierc': 3232, 'transport': 3233,
'sunshin': 3234, 'tomato': 3235, 'wallac': 3236, 'packag': 3237,
'marin': 3238, 'coloni': 3239, 'ought': 3240, 'hollow': 3241,
'juvenil': 3242, 'album': 3243, 'q': 3244, 'sixti': 3245, 'kenneth':
3246, 'contrari': 3247, 'evolv': 3248, 'choreographi': 3249, 'prom':

3250, 'donna': 3251, 'karen': 3252, 'pray': 3253, '2003': 3254,
'viciou': 3255, 'revolutionari': 3256, 'li': 3257, 'eyr': 3258,
'brazil': 3259, 'rambo': 3260, 'awak': 3261, 'horrid': 3262, 'saint':
3263, 'dose': 3264, 'ramon': 3265, 'conserv': 3266, 'kapoor': 3267,
'mummi': 3268, 'boast': 3269, 'defi': 3270, 'stole': 3271, 'twelv':
3272, 'beatti': 3273, 'nerd': 3274, 'overr': 3275, 'option': 3276,
'mildr': 3277, 'blade': 3278, 'candid': 3279, 'ireland': 3280,
'detract': 3281, 'astonish': 3282, 'funer': 3283, 'confirm': 3284,
'kirk': 3285, 'trio': 3286, 'protest': 3287, 'flame': 3288, 'jazz':
3289, 'collabor': 3290, 'global': 3291, 'fulci': 3292, 'natali': 3293,
'altman': 3294, 'yellow': 3295, 'bottl': 3296, 'racial': 3297, 'leap':
3298, 'tommi': 3299, 'shade': 3300, 'whip': 3301, 'enterpris': 3302,
'mystic': 3303, 'delici': 3304, 'bull': 3305, 'audio': 3306,
'destini': 3307, 'blake': 3308, 'nicholson': 3309, 'spit': 3310,
'adolesc': 3311, 'meaningless': 3312, 'fonda': 3313, 'enchant': 3314,
'swedish': 3315, 'pseudo': 3316, 'popcorn': 3317, 'threw': 3318,
'visibl': 3319, 'inherit': 3320, 'merci': 3321, 'vivid': 3322,
'altogeth': 3323, 'reunit': 3324, 'bedroom': 3325, 'harder': 3326,
'staff': 3327, 'todd': 3328, 'neo': 3329, 'synopsi': 3330, 'tip':
3331, 'await': 3332, 'wire': 3333, 'crocodil': 3334, 'reserv': 3335,
'suspici': 3336, 'fanat': 3337, 'uneven': 3338, 'exhibit': 3339,
'respond': 3340, 'leonard': 3341, 'decor': 3342, 'befriend': 3343,
'bust': 3344, 'ruthless': 3345, 'moodi': 3346, 'kennedi': 3347,
'lawrenc': 3348, 'madonna': 3349, 'roommat': 3350, 'voight': 3351,
'lemmon': 3352, 'atlanti': 3353, 'jew': 3354, 'edi': 3355, 'holli':
3356, 'bold': 3357, 'unsettl': 3358, 'chao': 3359, 'ventur': 3360,
'carl': 3361, 'dimens': 3362, 'bargain': 3363, 'centr': 3364, 'rural':
3365, 'abysm': 3366, 'voyag': 3367, 'bradi': 3368, 'garner': 3369,
'2007': 3370, 'audit': 3371, 'incident': 3372, 'clumsi': 3373,
'clint': 3374, 'palma': 3375, 'mall': 3376, 'versu': 3377, 'lit':
3378, 'trail': 3379, 'echo': 3380, 'timon': 3381, 'ant': 3382,
'nearbi': 3383, 'cuba': 3384, 'characterist': 3385, 'humili': 3386,
'imperson': 3387, 'daddi': 3388, 'tiger': 3389, 'hart': 3390, 'troop':
3391, '2nd': 3392, 'cd': 3393, 'elimin': 3394, 'poetic': 3395,
'immigr': 3396, 'cari': 3397, 'neglect': 3398, 'wealth': 3399,
'acknowledg': 3400, 'marshal': 3401, 'repuls': 3402, 'mickey': 3403,
'celluloid': 3404, 'solo': 3405, 'homer': 3406, 'domest': 3407,
'saga': 3408, 'paus': 3409, 'collaps': 3410, 'mistaken': 3411,
'jeffrey': 3412, 'pun': 3413, 'infect': 3414, 'prejudic': 3415, 'hbo':
3416, '1996': 3417, 'inan': 3418, 'promin': 3419, 'equip': 3420,
'inappropri': 3421, 'apolog': 3422, 'coat': 3423, 'interrupt': 3424,
'coffe': 3425, 'tribe': 3426, 'leon': 3427, 'chest': 3428, 'cake':
3429, 'undoubtedli': 3430, 'assembl': 3431, 'pant': 3432, 'harvey':
3433, 'milk': 3434, 'sore': 3435, 'gear': 3436, 'ginger': 3437,
'olivi': 3438, 'jenni': 3439, 'retain': 3440, 'embrac': 3441, 'exot':
3442, 'colleagu': 3443, 'brooklyn': 3444, 'highest': 3445, 'polanski':
3446, 'airplan': 3447, 'instant': 3448, 'trace': 3449, 'primari':
3450, 'consum': 3451, 'humbl': 3452, 'solut': 3453, 'institut': 3454,
'aveng': 3455, 'pen': 3456, 'pot': 3457, 'colonel': 3458, 'maggi':
3459, 'furthermor': 3460, 'devast': 3461, 'vulgar': 3462, 'florida':

3463, 'linda': 3464, 'dian': 3465, 'smaller': 3466, 'disabl': 3467, 'bowl': 3468, 'gender': 3469, 'principl': 3470, 'strain': 3471, 'illog': 3472, 'rick': 3473, 'poke': 3474, 'ya': 3475, 'dutch': 3476, 'cope': 3477, 'godzilla': 3478, '3rd': 3479, 'sale': 3480, 'seduct': 3481, 'outer': 3482, 'descend': 3483, '1999': 3484, 'wife': 3485, 'dud': 3486, 'rabbit': 3487, 'yard': 3488, 'dive': 3489, 'bubbl': 3490, 'gundam': 3491, 'lol': 3492, 'devoid': 3493, 'vast': 3494, 'scope': 3495, 'inferior': 3496, 'cue': 3497, 'primarili': 3498, 'mixtur': 3499, 'predecessor': 3500, 'blatant': 3501, 'hal': 3502, 'glamor': 3503, 'beneath': 3504, 'secondli': 3505, 'gloriou': 3506, 'et': 3507, 'senseless': 3508, 'shelf': 3509, 'hideou': 3510, 'countrysid': 3511, 'grinch': 3512, 'april': 3513, 'museum': 3514, 'shirley': 3515, 'arab': 3516, 'streep': 3517, 'alert': 3518, 'disjoint': 3519, 'domino': 3520, 'z': 3521, 'pearl': 3522, 'alfr': 3523, 'talki': 3524, 'aggress': 3525, 'myer': 3526, 'casual': 3527, 'invas': 3528, 'trademark': 3529, 'simplist': 3530, 'garbo': 3531, 'breed': 3532, 'hopeless': 3533, 'mail': 3534, 'illeg': 3535, 'acid': 3536, 'vanish': 3537, 'defens': 3538, 'mayor': 3539, 'unhappi': 3540, 'stir': 3541, 'maci': 3542, 'khan': 3543, 'hardcor': 3544, 'experiment': 3545, 'slide': 3546, 'loyal': 3547, 'robberi': 3548, 'sh': 3549, 'obtain': 3550, 'applaud': 3551, 'robinson': 3552, 'stack': 3553, 'rendit': 3554, 'stellar': 3555, 'uwe': 3556, 'boom': 3557, 'oz': 3558, 'disgrac': 3559, 'dismiss': 3560, 'amanda': 3561, 'counter': 3562, 'berlin': 3563, 'soccer': 3564, 'fri': 3565, 'tenant': 3566, 'incomprehens': 3567, 'wont': 3568, 'grandfath': 3569, 'topless': 3570, 'tempt': 3571, 'rifl': 3572, 'recruit': 3573, 'blew': 3574, 'span': 3575, 'emphasi': 3576, 'spider': 3577, 'diana': 3578, 'psychic': 3579, 'dicken': 3580, 'craze': 3581, 'scroog': 3582, 'declin': 3583, 'hartley': 3584, 'parad': 3585, 'revolt': 3586, 'goer': 3587, 'porno': 3588, 'shed': 3589, 'niro': 3590, 'ethnic': 3591, 'ration': 3592, 'sibl': 3593, 'woo': 3594, 'trashy': 3595, 'faster': 3596, 'resurrect': 3597, 'justin': 3598, 'wet': 3599, 'bitch': 3600, 'riot': 3601, 'lumet': 3602, 'sympath': 3603, 'intim': 3604, 'shaw': 3605, 'jonathan': 3606, 'nephew': 3607, '00': 3608, 'weakest': 3609, 'farmer': 3610, 'mario': 3611, 'region': 3612, 'unreal': 3613, 'ensur': 3614, 'partial': 3615, 'rider': 3616, 'eager': 3617, 'gap': 3618, 'patriot': 3619, 'commend': 3620, 'ballet': 3621, 'immort': 3622, 'hesit': 3623, 'wheel': 3624, 'hopper': 3625, 'andr': 3626, 'honesti': 3627, 'feminist': 3628, 'steam': 3629, 'dealer': 3630, 'worm': 3631, 'enlighten': 3632, 'slick': 3633, 'choru': 3634, 'wendi': 3635, 'biographi': 3636, 'lena': 3637, 'psychopath': 3638, 'nostalg': 3639, 'mutant': 3640, 'blunt': 3641, 'prequel': 3642, 'properti': 3643, 'similarli': 3644, 'wore': 3645, 'sappi': 3646, 'morri': 3647, 'leo': 3648, 'snap': 3649, 'confin': 3650, 'safeti': 3651, 'charlott': 3652, 'kingdom': 3653, 'sandra': 3654, 'hung': 3655, 'owen': 3656, 'skull': 3657, 'repress': 3658, 'victori': 3659, 'util': 3660, 'composit': 3661, 'macarthur': 3662, 'vice': 3663, 'franco': 3664, 'compens': 3665, 'nervou': 3666, '1972': 3667, 'acquir': 3668, 'bow': 3669, 'whoopi': 3670, 'miseri': 3671, 'drain': 3672, 'bumbl': 3673, 'campbel': 3674, 'tad': 3675,

'drum': 3676, 'recycl': 3677, 'pattern': 3678, 'tail': 3679,
'heartbreak': 3680, 'exit': 3681, 'rocki': 3682, 'valuabl': 3683,
'dalton': 3684, 'del': 3685, 'bonu': 3686, 'repli': 3687, 'dust':
3688, 'cg': 3689, 'speci': 3690, 'strand': 3691, 'thru': 3692, 'rope':
3693, 'compass': 3694, 'rambl': 3695, 'despair': 3696, 'hyde': 3697,
'bergman': 3698, 'montana': 3699, 'kyle': 3700, 'farrel': 3701,
'latin': 3702, 'emperor': 3703, 'deed': 3704, 'snl': 3705, 'radic':
3706, 'orson': 3707, 'tonight': 3708, 'roth': 3709, 'contempl': 3710,
'pour': 3711, '35': 3712, 'gal': 3713, 'romp': 3714, 'carradin': 3715,
'mistress': 3716, 'percept': 3717, 'bleed': 3718, 'airport': 3719,
'downhil': 3720, 'slug': 3721, 'rotten': 3722, 'rapist': 3723, 'da':
3724, 'martian': 3725, 'oppress': 3726, 'gimmick': 3727, 'wacki':
3728, 'chess': 3729, 'olli': 3730, 'champion': 3731, 'dazzl': 3732,
'programm': 3733, 'arc': 3734, 'taught': 3735, 'slash': 3736,
'attorney': 3737, 'arguabl': 3738, 'pervers': 3739, 'pervert': 3740,
'mislead': 3741, 'preach': 3742, 'paltrow': 3743, 'belt': 3744,
'pursuit': 3745, 'stilt': 3746, 'unpredict': 3747, 'banal': 3748,
'1983': 3749, 'heal': 3750, 'edgar': 3751, 'tooth': 3752, 'melodi':
3753, 'tackl': 3754, 'shelley': 3755, 'raymond': 3756, 'cleverli':
3757, 'virginia': 3758, 'vocal': 3759, 'passeng': 3760, 'gambl': 3761,
'vengeanc': 3762, 'orang': 3763, 'mesmer': 3764, 'graham': 3765,
'plight': 3766, 'sensat': 3767, 'franki': 3768, 'closest': 3769,
'conneri': 3770, 'employe': 3771, 'maid': 3772, 'poem': 3773, 'duval':
3774, 'tiresom': 3775, 'uplift': 3776, 'marti': 3777, 'bela': 3778,
'chicken': 3779, 'rubi': 3780, 'dixon': 3781, 'inject': 3782,
'amitabh': 3783, 'quarter': 3784, 'swallow': 3785, 'abraham': 3786,
'scottish': 3787, 'paranoia': 3788, 'calm': 3789, 'climact': 3790,
'yawn': 3791, 'iran': 3792, 'extens': 3793, 'habit': 3794, 'numb':
3795, 'pokemon': 3796, 'outing': 3797, 'secretli': 3798,
'convincingli': 3799, 'suffic': 3800, 'whine': 3801, 'monologu': 3802,
'giggl': 3803, 'mute': 3804, 'profan': 3805, 'crystal': 3806, '1968':
3807, 'engross': 3808, 'volum': 3809, 'tube': 3810, 'clone': 3811,
'bay': 3812, 'gerard': 3813, 'sirk': 3814, 'lundgren': 3815, 'junior':
3816, 'lowest': 3817, 'plod': 3818, 'linger': 3819, 'bend': 3820,
'chicago': 3821, 'richardson': 3822, 'nichola': 3823, 'fed': 3824,
'meander': 3825, 'spock': 3826, 'abort': 3827, 'expand': 3828,
'backward': 3829, 'ethan': 3830, 'dispos': 3831, 'earl': 3832,
'profess': 3833, 'poetri': 3834, 'trend': 3835, 'underst': 3836,
'septemb': 3837, 'franci': 3838, 'surpass': 3839, 'taxi': 3840, 'im':
3841, 'grotesqu': 3842, 'austen': 3843, 'frankenstein': 3844,
'compliment': 3845, 'waitress': 3846, 'nostalgia': 3847, 'greedi':
3848, 'sue': 3849, 'instrument': 3850, 'rubber': 3851, 'rant': 3852,
'spoke': 3853, 'simplic': 3854, 'muddl': 3855, 'eugen': 3856,
'household': 3857, 'tourist': 3858, 'literatur': 3859, 'lure': 3860,
'hum': 3861, 'catchi': 3862, 'stallon': 3863, 'cannon': 3864, 'der':
3865, 'myth': 3866, 'mundan': 3867, 'descent': 3868, 'econom': 3869,
'dysfunct': 3870, 'map': 3871, 'duck': 3872, 'irrelev': 3873,
'occupi': 3874, 'molli': 3875, 'lang': 3876, 'cent': 3877, 'dement':
3878, 'alongsid': 3879, 'stale': 3880, 'eaten': 3881, 'crucial': 3882,
'phantom': 3883, 'mortal': 3884, 'flee': 3885, 'dictat': 3886,

'recognit': 3887, 'carel': 3888, 'insur': 3889, 'deaf': 3890, 'june': 3891, 'mankind': 3892, 'bacal': 3893, 'phoni': 3894, 'louis': 3895, 'damon': 3896, 'furi': 3897, 'recognis': 3898, 'hello': 3899, 'equival': 3900, 'omen': 3901, 'firstli': 3902, 'randi': 3903, 'coast': 3904, 'sissi': 3905, 'daisi': 3906, 'twilight': 3907, 'antwon': 3908, 'reign': 3909, 'distinguish': 3910, 'onlin': 3911, 'rude': 3912, 'grayson': 3913, 'lengthi': 3914, 'rooney': 3915, 'loyalti': 3916, 'wisdom': 3917, 'dreari': 3918, 'heel': 3919, 'blackmail': 3920, 'bump': 3921, 'newli': 3922, '1973': 3923, 'labor': 3924, 'ashley': 3925, 'freez': 3926, 'cyborg': 3927, 'buffalo': 3928, 'likewis': 3929, 'drake': 3930, 'bike': 3931, 'biko': 3932, 'damm': 3933, 'basketbal': 3934, 'approv': 3935, 'pink': 3936, 'exposur': 3937, 'baddi': 3938, 'worn': 3939, 'unorigin': 3940, 'startl': 3941, 'chronicl': 3942, 'barrymor': 3943, 'sailor': 3944, 'tunnel': 3945, 'nineti': 3946, 'inher': 3947, 'keith': 3948, 'provoc': 3949, 'incorpor': 3950, 'interior': 3951, 'ridden': 3952, 'boxer': 3953, 'proce': 3954, 'attribut': 3955, 'emphas': 3956, 'analysi': 3957, 'vein': 3958, 'prey': 3959, 'butler': 3960, 'stalker': 3961, 'meyer': 3962, 'mighti': 3963, 'degrad': 3964, 'robbin': 3965, 'meg': 3966, 'bunni': 3967, 'indiffer': 3968, 'belushi': 3969, 'hypnot': 3970, 'undeni': 3971, 'barrel': 3972, 'substitut': 3973, 'underli': 3974, 'drift': 3975, 'simmon': 3976, 'walsh': 3977, 'improvis': 3978, 'er': 3979, 'othello': 3980, 'fleet': 3981, 'predat': 3982, 'nicol': 3983, 'unrel': 3984, 'millionair': 3985, 'carla': 3986, 'elm': 3987, 'condemn': 3988, 'julian': 3989, 'mormon': 3990, 'exquisit': 3991, 'nyc': 3992, 'reid': 3993, 'alison': 3994, 'disord': 3995, 'agenda': 3996, 'roof': 3997, 'warmth': 3998, 'watson': 3999, 'hay': 4000, 'edgi': 4001, 'enthusiasm': 4002, 'lampoon': 4003, 'vital': 4004, 'unawar': 4005, 'errol': 4006, 'marion': 4007, 'shove': 4008, 'mtv': 4009, '3d': 4010, 'palac': 4011, 'rukh': 4012, 'novak': 4013, 'firm': 4014, 'priceless': 4015, 'alarm': 4016, 'greed': 4017, 'dolph': 4018, 'gestur': 4019, 'session': 4020, 'peril': 4021, 'eastern': 4022, 'orlean': 4023, 'beatl': 4024, 'cassidi': 4025, 'showdown': 4026, 'iraq': 4027, 'angela': 4028, 'randomli': 4029, 'petti': 4030, 'profit': 4031, 'what': 4032, '13th': 4033, 'ponder': 4034, 'simultan': 4035, 'unleash': 4036, 'preserv': 4037, 'campaign': 4038, '1933': 4039, 'distort': 4040, 'pamela': 4041, 'zizek': 4042, 'testament': 4043, 'glanc': 4044, 'thompson': 4045, 'sergeant': 4046, 'coup': 4047, 'crown': 4048, 'israel': 4049, 'peck': 4050, 'minimum': 4051, 'drip': 4052, 'nun': 4053, 'valentin': 4054, 'spain': 4055, 'championship': 4056, 'travesti': 4057, '1984': 4058, 'reson': 4059, 'han': 4060, 'scotland': 4061, 'contradict': 4062, 'empathi': 4063, 'restrain': 4064, 'valley': 4065, 'cream': 4066, 'crawl': 4067, 'calib': 4068, 'perpetu': 4069, 'brenda': 4070, 'shootout': 4071, 'cooki': 4072, 'stake': 4073, 'crow': 4074, 'sabrina': 4075, 'realm': 4076, 'mon': 4077, 'represent': 4078, 'exposit': 4079, 'regist': 4080, 'gentleman': 4081, 'bro': 4082, 'miyazaki': 4083, 'unimagin': 4084, 'din': 4085, 'rout': 4086, 'climat': 4087, 'wig': 4088, 'buster': 4089, 'stroke': 4090, 'jan': 4091, 'quinn': 4092, 'kurosawa': 4093, 'fido': 4094, 'pretens': 4095, 'distress': 4096, 'darren': 4097,

'wax': 4098, 'passabl': 4099, 'absent': 4100, 'businessman': 4101, 'meryl': 4102, 'geek': 4103, 'derang': 4104, '1987': 4105, 'femm': 4106, 'pole': 4107, 'josh': 4108, 'sammi': 4109, 'monoton': 4110, 'demis': 4111, 'compromis': 4112, 'sucker': 4113, '1997': 4114, 'shaki': 4115, 'censor': 4116, 'spacey': 4117, 'traumat': 4118, 'abomin': 4119, 'perceiv': 4120, 'painter': 4121, 'ross': 4122, 'crawford': 4123, 'greg': 4124, 'unseen': 4125, 'delic': 4126, 'fuller': 4127, 'cloud': 4128, 'tacki': 4129, 'stargat': 4130, 'shoddi': 4131, 'mclaglen': 4132, 'soderbergh': 4133, 'warrant': 4134, 'unsatisfi': 4135, 'baldwin': 4136, 'dana': 4137, 'ustinov': 4138, 'tarantino': 4139, 'clash': 4140, 'seal': 4141, 'exclus': 4142, 'nathan': 4143, 'verbal': 4144, 'deniro': 4145, 'antonioni': 4146, 'polici': 4147, 'reluct': 4148, 'sid': 4149, 'judgment': 4150, 'fenc': 4151, 'accuraci': 4152, 'click': 4153, 'wholli': 4154, 'jewel': 4155, 'uncov': 4156, '1993': 4157, 'anchor': 4158, 'valid': 4159, 'dee': 4160, 'norm': 4161, 'primit': 4162, 'expedit': 4163, 'austin': 4164, 'furiou': 4165, 'fog': 4166, 'tech': 4167, 'deceas': 4168, 'kumar': 4169, 'unravel': 4170, 'correctli': 4171, 'darn': 4172, 'pocket': 4173, '1971': 4174, 'joel': 4175, 'sunni': 4176, 'shanghai': 4177, 'fought': 4178, '2008': 4179, 'seldom': 4180, 'unfair': 4181, 'temper': 4182, '3000': 4183, 'conduct': 4184, 'murray': 4185, 'nicola': 4186, 'sustain': 4187, 'enforc': 4188, 'alec': 4189, 'tax': 4190, 'hallucin': 4191, 'fart': 4192, 'crippl': 4193, 'fabric': 4194, 'malon': 4195, 'sheet': 4196, 'roller': 4197, 'sand': 4198, 'patienc': 4199, 'bake': 4200, 'dreck': 4201, '1995': 4202, 'behold': 4203, 'debt': 4204, 'wang': 4205, 'logan': 4206, 'trait': 4207, 'wretch': 4208, 'ritual': 4209, 'vanc': 4210, 'mode': 4211, 'slam': 4212, 'clerk': 4213, 'despis': 4214, 'canyon': 4215, 'phil': 4216, 'stark': 4217, 'runner': 4218, 'schedul': 4219, 'clau': 4220, 'rita': 4221, 'legaci': 4222, 'pete': 4223, 'technicolor': 4224, 'shell': 4225, 'guitar': 4226, 'outlin': 4227, 'fundament': 4228, 'tactic': 4229, 'critiqu': 4230, 'divid': 4231, 'exhaust': 4232, 'bias': 4233, 'robber': 4234, 'soup': 4235, 'isabel': 4236, 'bridget': 4237, 'squad': 4238, 'conscious': 4239, 'grief': 4240, 'sweep': 4241, 'scriptwrit': 4242, 'preposter': 4243, 'helpless': 4244, 'stuart': 4245, 'penni': 4246, 'preston': 4247, 'rear': 4248, 'restrict': 4249, 'lacklust': 4250, 'implic': 4251, 'russia': 4252, 'newman': 4253, 'connor': 4254, 'liberti': 4255, 'consciou': 4256, 'sniper': 4257, 'cigarette': 4258, 'marc': 4259, 'invad': 4260, 'flair': 4261, 'boyl': 4262, 'kansa': 4263, 'alley': 4264, 'downey': 4265, 'culmin': 4266, 'jodi': 4267, 'vomit': 4268, 'sugar': 4269, 'unexpectedli': 4270, 'inabl': 4271, 'passag': 4272, 'drove': 4273, 'sentinel': 4274, 'agenc': 4275, 'rehash': 4276, 'delv': 4277, 'bloom': 4278, 'alicia': 4279, 'gregori': 4280, 'propos': 4281, 'jacket': 4282, 'palanc': 4283, 'rehears': 4284, 'pale': 4285, 'cap': 4286, 'kolchak': 4287, '22': 4288, 'rod': 4289, 'mccoy': 4290, 'horn': 4291, 'lush': 4292, 'tripe': 4293, 'asylum': 4294, 'rampag': 4295, 'aesthet': 4296, 'bacon': 4297, 'behaviour': 4298, 'feat': 4299, '1936': 4300, 'sharon': 4301, 'karl': 4302, 'delet': 4303, 'tendenc': 4304, 'awhil': 4305, 'yeti': 4306, 'wrench': 4307, 'ladder': 4308, 'vet': 4309, 'foxx': 4310, 'arrow':

4311, 'chainsaw': 4312, 'improb': 4313, 'hungri': 4314, 'sung': 4315,
'lurk': 4316, 'thunderbird': 4317, 'rumor': 4318, 'financ': 4319,
'stream': 4320, 'paradis': 4321, '1920': 4322, 'filler': 4323,
'tomorrow': 4324, 'amazon': 4325, 'basing': 4326, 'spice': 4327,
'paramount': 4328, 'weav': 4329, 'wagner': 4330, '1978': 4331,
'rhythm': 4332, 'conscienc': 4333, 'wildli': 4334, 'suspicion': 4335,
'tasteless': 4336, 'scoop': 4337, 'underneath': 4338, 'aristocrat':
4339, 'globe': 4340, '1988': 4341, 'loneli': 4342, 'coaster': 4343,
'fright': 4344, 'shortcom': 4345, 'hulk': 4346, '19th': 4347,
'suffici': 4348, 'minu': 4349, 'prank': 4350, 'elit': 4351, 'newcom':
4352, 'el': 4353, 'visitor': 4354, 'hackney': 4355, 'choppi': 4356,
'posey': 4357, 'abrupt': 4358, 'grudg': 4359, 'paxton': 4360,
'standout': 4361, 'quietli': 4362, 'en': 4363, 'ram': 4364, 'iv':
4365, '75': 4366, 'rub': 4367, 'naughti': 4368, 'entranc': 4369,
'counterpart': 4370, 'atroc': 4371, 'straightforward': 4372, 'dirt':
4373, 'worship': 4374, 'penn': 4375, 'springer': 4376, 'smell': 4377,
'brit': 4378, 'chamberlain': 4379, '1939': 4380, 'immers': 4381,
'lectur': 4382, 'couch': 4383, 'heist': 4384, 'leigh': 4385,
'literari': 4386, '1989': 4387, 'inmat': 4388, 'minist': 4389,
'cancer': 4390, 'curli': 4391, 'teas': 4392, 'ingeni': 4393, 'bread':
4394, 'recogniz': 4395, 'secondari': 4396, 'impos': 4397, 'beverli':
4398, 'wwe': 4399, 'tierney': 4400, 'hopkin': 4401, 'chavez': 4402,
'enthrall': 4403, '1986': 4404, 'injuri': 4405, 'sassi': 4406,
'variat': 4407, 'yearn': 4408, 'nemesi': 4409, 'heartfelt': 4410,
'clan': 4411, 'geni': 4412, 'skeptic': 4413, 'missil': 4414, 'net':
4415, 'entitl': 4416, 'misguid': 4417, 'moreov': 4418, 'cattl': 4419,
'convert': 4420, 'ace': 4421, 'lindsay': 4422, 'watcher': 4423,
'bernard': 4424, 'policeman': 4425, 'transcend': 4426, 'esther': 4427,
'morbid': 4428, 'duel': 4429, 'ratso': 4430, 'sublim': 4431, 'quaid':
4432, 'attenborough': 4433, 'laurenc': 4434, 'nolan': 4435, 'bye':
4436, 'tyler': 4437, 'graini': 4438, 'obstacl': 4439, 'out': 4440,
'brood': 4441, 'carlito': 4442, 'setup': 4443, 'characteris': 4444,
'facil': 4445, 'uncut': 4446, 'hk': 4447, 'vader': 4448, 'puppi':
4449, 'cruelti': 4450, 'mytholog': 4451, '1979': 4452, 'buzz': 4453,
'hopelessli': 4454, 'bean': 4455, 'spiral': 4456, 'reliabl': 4457,
'moder': 4458, 'unexplain': 4459, 'grin': 4460, 'poe': 4461, 'steady':
4462, 'kitti': 4463, 'egg': 4464, 'enabl': 4465, 'youngest': 4466,
'diari': 4467, 'artsi': 4468, 'dont': 4469, 'rosemari': 4470,
'kidman': 4471, 'disastr': 4472, 'baffl': 4473, 'heap': 4474,
'effici': 4475, 'decept': 4476, 'acquaint': 4477, 'bounc': 4478,
'weather': 4479, 'despic': 4480, 'niec': 4481, 'underworld': 4482,
'narrow': 4483, 'exterior': 4484, 'gina': 4485, 'spontan': 4486,
'1969': 4487, 'sweat': 4488, 'preming': 4489, 'brendan': 4490,
'hammi': 4491, 'athlet': 4492, 'clueless': 4493, 'kline': 4494,
'fuel': 4495, 'patricia': 4496, 'martha': 4497, 'beware': 4498,
'gillian': 4499, 'oblig': 4500, 'hain': 4501, 'bronson': 4502,
'christin': 4503, 'biker': 4504, 'sleepwalk': 4505, 'harmless': 4506,
'scar': 4507, 'injur': 4508, 'insipid': 4509, 'dilemma': 4510,
'mayhem': 4511, 'angst': 4512, 'mermaid': 4513, 'circu': 4514, 'goof':
4515, 'hepburn': 4516, 'loath': 4517, 'astound': 4518, 'outlaw': 4519,

'uh': 4520, 'housewif': 4521, 'loi': 4522, 'enlist': 4523, 'preachi': 4524, '19': 4525, 'tick': 4526, 'lester': 4527, 'fontain': 4528, 'viewpoint': 4529, 'rome': 4530, 'sooner': 4531, 'trigger': 4532, 'taboo': 4533, 'renaiss': 4534, 'suprem': 4535, 'virtu': 4536, 'candl': 4537, '73': 4538, 'analyz': 4539, 'shatter': 4540, 'dandi': 4541, 'headach': 4542, 'surgeri': 4543, 'amor': 4544, 'stair': 4545, 'hokey': 4546, 'boston': 4547, 'sox': 4548, 'dismal': 4549, 'contempt': 4550, 'glorifi': 4551, 'dish': 4552, 'foolish': 4553, 'spade': 4554, 'oldest': 4555, 'hostag': 4556, 'cassavet': 4557, 'fluff': 4558, 'intric': 4559, 'guin': 4560, 'scorses': 4561, 'filth': 4562, 'hooker': 4563, 'macho': 4564, 'camcord': 4565, 'claustrophob': 4566, 'corbett': 4567, 'overlong': 4568, 'redund': 4569, 'whore': 4570, 'idol': 4571, 'salt': 4572, 'bent': 4573, 'gere': 4574, 'immatur': 4575, 'slimi': 4576, 'ariel': 4577, 'zoom': 4578, 'tripl': 4579, 'phenomenon': 4580, 'stimul': 4581, 'steer': 4582, 'ebert': 4583, 'fascist': 4584, 'shield': 4585, 'preced': 4586, 'transplant': 4587, 'dwarf': 4588, 'harold': 4589, 'widescreen': 4590, 'cush': 4591, 'down': 4592, 'proport': 4593, 'flirt': 4594, 'beard': 4595, 'antagonist': 4596, 'keen': 4597, 'gasp': 4598, 'flag': 4599, 'radiat': 4600, 'astronaut': 4601, 'shred': 4602, 'frantic': 4603, 'strongest': 4604, 'cow': 4605, 'margin': 4606, 'joker': 4607, 'zane': 4608, '1981': 4609, 'mount': 4610, 'spree': 4611, '1976': 4612, 'messi': 4613, 'remad': 4614, 'naschi': 4615, 'mutual': 4616, 'gabl': 4617, 'perman': 4618, 'trivia': 4619, 'muscl': 4620, 'faint': 4621, 'flashi': 4622, 'rhyme': 4623, 'cohen': 4624, 'nolt': 4625, 'assert': 4626, 'conquer': 4627, 'corman': 4628, 'schlock': 4629, 'spinal': 4630, 'obligatori': 4631, 'alvin': 4632, 'www': 4633, 'scandal': 4634, 'vaniti': 4635, 'interestingly': 4636, 'persuad': 4637, 'boob': 4638, 'carey': 4639, 'info': 4640, 'sensual': 4641, 'brush': 4642, 'off': 4643, 'bitten': 4644, 'divin': 4645, 'deer': 4646, 'instruct': 4647, 'claud': 4648, 'ritchi': 4649, 'fishburn': 4650, '1945': 4651, '28': 4652, 'discern': 4653, 'mobil': 4654, 'wield': 4655, 'bachelor': 4656, 'repris': 4657, 'flock': 4658, '95': 4659, 'strive': 4660, 'inflict': 4661, 'triangl': 4662, 'hara': 4663, 'resum': 4664, 'departur': 4665, 'raj': 4666, 'someday': 4667, 'barn': 4668, 'danish': 4669, 'neurot': 4670, 'mol': 4671, 'aborigin': 4672, 'archiv': 4673, 'senior': 4674, 'dylan': 4675, 'pacif': 4676, 'heartwarm': 4677, 'vibrant': 4678, 'casino': 4679, 'hilar': 4680, 'carlo': 4681, 'mobster': 4682, 'rot': 4683, 'luka': 4684, 'hug': 4685, 'cb': 4686, 'undermin': 4687, 'cher': 4688, 'kathryn': 4689, 'wendigo': 4690, 'miracul': 4691, 'neill': 4692, 'submit': 4693, 'timberlak': 4694, 'bate': 4695, 'cliffhang': 4696, 'harrison': 4697, 'dame': 4698, 'dim': 4699, 'colin': 4700, 'earnest': 4701, 'proclaim': 4702, 'banter': 4703, 'helm': 4704, 'fragil': 4705, 'hapless': 4706, 'recit': 4707, 'pixar': 4708, 'artwork': 4709, 'ish': 4710, 'biblic': 4711, 'cycl': 4712, 'prophet': 4713, 'melissa': 4714, 'loretta': 4715, 'jade': 4716, 'axe': 4717, 'europa': 4718, 'frontier': 4719, 'anton': 4720, 'traffic': 4721, 'clad': 4722, 'pickford': 4723, 'parson': 4724, 'articl': 4725, 'http': 4726, 'orphan': 4727, 'alexandr': 4728, 'razor': 4729, 'cerebr': 4730, 'vile': 4731, 'toronto': 4732, 'pc': 4733, 'token':

4734, 'shepherd': 4735, 'legitim': 4736, 'redneck': 4737, 'jo': 4738, 'vanessa': 4739, 'misfortun': 4740, 'northern': 4741, 'lucil': 4742, 'mason': 4743, 'feast': 4744, 'flavor': 4745, 'bikini': 4746, 'breakfast': 4747, 'lui': 4748, 'winchest': 4749, 'foil': 4750, 'eli': 4751, 'holocaust': 4752, 'sicken': 4753, 'electron': 4754, 'marlon': 4755, 'static': 4756, 'bondag': 4757, 'illus': 4758, 'wardrob': 4759, 'venom': 4760, 'blatantli': 4761, 'akin': 4762, 'seedi': 4763, 'jordan': 4764, 'trier': 4765, 'rooki': 4766, 'antholog': 4767, 'aris': 4768, 'choke': 4769, 'nope': 4770, 'milo': 4771, 'estrang': 4772, 'mathieu': 4773, 'uma': 4774, 'isra': 4775, 'clinic': 4776, 'cartoonish': 4777, 'styliz': 4778, 'feminin': 4779, 'smack': 4780, 'knightley': 4781, 'turd': 4782, 'frog': 4783, 'ceremoni': 4784, 'retriev': 4785, 'dudley': 4786, 'deem': 4787, 'oppon': 4788, 'leather': 4789, 'linear': 4790, 'audrey': 4791, 'disregard': 4792, 'psych': 4793, 'tack': 4794, 'affleck': 4795, 'peer': 4796, 'outdat': 4797, 'shorter': 4798, 'howl': 4799, 'nightclub': 4800, 'comprehend': 4801, 'abund': 4802, 'swept': 4803, 'gilbert': 4804, 'wrestler': 4805, 'magician': 4806, 'glare': 4807, 'fifth': 4808, 'charlton': 4809, 'gunga': 4810, 'ideolog': 4811, 'boyer': 4812, 'highway': 4813, 'huston': 4814, 'braveheart': 4815, 'snatch': 4816, '4th': 4817, 'cuban': 4818, 'salman': 4819, 'lavish': 4820, 'collector': 4821, 'conrad': 4822, 'einstein': 4823, 'corn': 4824, 'lifeless': 4825, 'btw': 4826, 'durat': 4827, 'toe': 4828, 'greet': 4829, 'plate': 4830, 'senat': 4831, 'energet': 4832, 'spawn': 4833, 'lighter': 4834, 'whack': 4835, 'evolut': 4836, 'phenomen': 4837, 'summar': 4838, 'sleaz': 4839, '1991': 4840, 'compris': 4841, 'monument': 4842, 'moe': 4843, 'boo': 4844, 'potter': 4845, '1994': 4846, 'newer': 4847, 'deliver': 4848, 'bastard': 4849, 'tara': 4850, 'spine': 4851, 'breakdown': 4852, 'client': 4853, 'bogu': 4854, 'uniformli': 4855, 'chip': 4856, 'cemeteri': 4857, 'mitch': 4858, 'goldsworthi': 4859, 'belli': 4860, 'trauma': 4861, 'evelyn': 4862, 'healthi': 4863, '1974': 4864, 'kent': 4865, 'appl': 4866, 'ol': 4867, 'jam': 4868, 'replay': 4869, 'embark': 4870, 'constitut': 4871, 'jule': 4872, 'jedi': 4873, 'bulk': 4874, 'luxuri': 4875, 'sorrow': 4876, 'signal': 4877, 'fluid': 4878, 'nina': 4879, 'undertak': 4880, 'firmli': 4881, 'historian': 4882, 'judd': 4883, 'ie': 4884, 'eleven': 4885, 'mcqueen': 4886, 'alleg': 4887, 'creek': 4888, 'clara': 4889, 'pronounc': 4890, 'capot': 4891, 'randolph': 4892, 'inaccuraci': 4893, 'spectacl': 4894, 'liu': 4895, '1977': 4896, 'bori': 4897, 'wtf': 4898, 'armstrong': 4899, 'neatli': 4900, 'jare': 4901, 'undead': 4902, 'outright': 4903, 'kazan': 4904, 'gilliam': 4905, 'cecil': 4906, 'occup': 4907, 'lex': 4908, 'rosario': 4909, 'poker': 4910, 'cape': 4911, 'unattract': 4912, 'palm': 4913, '1985': 4914, 'miami': 4915, 'comb': 4916, 'kiddi': 4917, 'carmen': 4918, 'id': 4919, 'vignett': 4920, 'relentless': 4921, 'meal': 4922, 'mum': 4923, 'inclus': 4924, 'subtli': 4925, 'ash': 4926, 'tokyo': 4927, 'groan': 4928, 'inaccur': 4929, 'bait': 4930, 'congratul': 4931, 'antonio': 4932, 'curtain': 4933, 'paula': 4934, 'lanc': 4935, 'aussi': 4936, 'unsuspect': 4937, 'galaxi': 4938, 'blur': 4939, 'basket': 4940, 'lauren': 4941, 'knee': 4942, 'propheci': 4943, 'decapit': 4944, 'pepper': 4945, 'forgiven':

```

4946, 'roar': 4947, 'sacrif': 4948, 'walt': 4949, 'porter': 4950,
'comprehens': 4951, 'pioneer': 4952, 'miniseri': 4953, 'conan': 4954,
'fruit': 4955, 'sidewalk': 4956, 'vain': 4957, 'truman': 4958,
'spray': 4959, 'abound': 4960, 'genet': 4961, 'goldblum': 4962, 'bsg':
4963, 'asset': 4964, 'evan': 4965, 'hackman': 4966, 'ghetto': 4967,
'assort': 4968, 'modest': 4969, 'omin': 4970, 'jill': 4971, 'macabr':
4972, 'ingrid': 4973, 'scarfac': 4974, 'dubiou': 4975, 'incorrect':
4976, 'epitom': 4977, 'orchestr': 4978, 'scariest': 4979, 'spill':
4980, 'sparkl': 4981, 'drone': 4982, 'victorian': 4983, 'detach':
4984, 'substanti': 4985, 'handicap': 4986, 'monti': 4987, 'playboy':
4988, 'optimist': 4989, 'frontal': 4990, 'sophi': 4991, 'rapidli':
4992, 'reincarn': 4993, 'verg': 4994, 'hostil': 4995, 'casper': 4996,
'masterson': 4997, 'weari': 4998, 'vastli': 4999}\n"

```

```

    ]
  },
  "source": [
    "word_dict = build_dict(train_X,5000,0)\n",
    "print(word_dict)"
  ],
  {
    "cell_type": "markdown",
    "metadata": {},
    "source": [
      "**Question:** What are the five most frequently appearing
(tokenized) words in the training set? Does it makes sense that these
words appear frequently in the training set?"
    ]
  },

```

```

  {
    "cell_type": "markdown",
    "metadata": {},
    "source": [
      "**Answer:**\n",
      "'movi'\n",
      "'film'\n",
      "'one'\n",
      "'like'\n",
      "'time'\n",
      "Yes, it makes sense, because they are common in reviews of
movies.\n"
    ]
  },

```

```

  {
    "cell_type": "code",
    "execution_count": 82,
    "metadata": {},
    "outputs": [
      {

```

```

    "data": {
      "text/plain": [
        "['movi', 'film', 'one', 'like', 'time']"
      ]
    },
    "execution_count": 82,
    "metadata": {},
    "output_type": "execute_result"
  }
],
"source": [
  "sorted_words = build_dict(train_X,5000,1)\n",
  "sorted_words[:5]"
]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "### Save `word_dict`\n",
    "\n",
    "Later on when we construct an endpoint which processes a
    submitted review we will need to make use of the `word_dict` which we
    have created. As such, we will save it to a file now for future use."
  ]
},
{
  "cell_type": "code",
  "execution_count": 83,
  "metadata": {},
  "outputs": [],
  "source": [
    "data_dir = '../data/pytorch' # The folder we will use for storing
    data\n",
    "if not os.path.exists(data_dir): # Make sure that the folder
    exists\n",
    "    os.makedirs(data_dir)"
  ]
},
{
  "cell_type": "code",
  "execution_count": 84,
  "metadata": {},
  "outputs": [],
  "source": [
    "with open(os.path.join(data_dir, 'word_dict.pkl'), \"wb\") as f:
    \n",
    "    pickle.dump(word_dict, f)"
  ]
},

```

```

{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "### Transform the reviews\n",
    "\n",
    "Now that we have our word dictionary which allows us to transform
the words appearing in the reviews into integers, it is time to make
use of it and convert our reviews to their integer sequence
representation, making sure to pad or truncate to a fixed length,
which in our case is `500`."
  ]
},
{
  "cell_type": "code",
  "execution_count": 85,
  "metadata": {},
  "outputs": [],
  "source": [
    "def convert_and_pad(word_dict, sentence, pad=500):\n",
    "    NOWORD = 0 # We will use 0 to represent the 'no word'
category\n",
    "    INFREQ = 1 # and we use 1 to represent the infrequent words,
i.e., words not appearing in word_dict\n",
    "    \n",
    "    working_sentence = [NOWORD] * pad\n",
    "    \n",
    "    for word_index, word in enumerate(sentence[:pad]):\n",
    "        if word in word_dict:\n",
    "            working_sentence[word_index] = word_dict[word]\n",
    "        else:\n",
    "            working_sentence[word_index] = INFREQ\n",
    "    \n",
    "    return working_sentence, min(len(sentence), pad)\n",
    "\n",
    "def convert_and_pad_data(word_dict, data, pad=500):\n",
    "    result = []\n",
    "    lengths = []\n",
    "    \n",
    "    for sentence in data:\n",
    "        converted, leng = convert_and_pad(word_dict, sentence,
pad)\n",
    "        result.append(converted)\n",
    "        lengths.append(leng)\n",
    "    \n",
    "    return np.array(result), np.array(lengths)"
  ]
},
{
  "cell_type": "code",

```

[illegible]

[illegible]

```

    ]
  }
],
"source": [
  "# Use this cell to examine one of the processed reviews to make
  sure everything is working as intended.\n",
  "print(\"len of a review\",len(train_X[333]))\n",
  "print(train_X[333])"
]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "**Question:** In the cells above we use the `preprocess_data` and
    `convert_and_pad_data` methods to process both the training and
    testing set. Why or why not might this be a problem?"
  ]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "**Answer:**\n",
    "There is no problem because both training and testing data is
    pre-processed the same way and there is less data leakage since the
    training data makes up the vocabulary."
  ]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "## Step 3: Upload the data to S3\n",
    "\n",
    "As in the XGBoost notebook, we will need to upload the training
    dataset to S3 in order for our training code to access it. For now we
    will save it locally and we will upload to S3 later on.\n",
    "\n",
    "### Save the processed training dataset locally\n",
    "\n",
    "It is important to note the format of the data that we are saving
    as we will need to know it when we write the training code. In our
    case, each row of the dataset has the form `label`, `length`,
    `review[500]` where `review[500]` is a sequence of `500` integers
    representing the words in the review."
  ]
},
{
  "cell_type": "code",

```

```

    "execution_count": 87,
    "metadata": {},
    "outputs": [],
    "source": [
        "import pandas as pd\n",
        "\n",
        "pd.concat([pd.DataFrame(train_y), pd.DataFrame(train_X_len),
pd.DataFrame(train_X)], axis=1) \\n",
        "        .to_csv(os.path.join(data_dir, 'train.csv'),
header=False, index=False)"
    ]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "### Uploading the training data\n",
        "\n",
        "\n",
        "Next, we need to upload the training data to the SageMaker
default S3 bucket so that we can provide access to it while training
our model."
    ]
},
{
    "cell_type": "code",
    "execution_count": 88,
    "metadata": {},
    "outputs": [],
    "source": [
        "import sagemaker\n",
        "\n",
        "sagemaker_session = sagemaker.Session()\n",
        "\n",
        "bucket = sagemaker_session.default_bucket()\n",
        "prefix = 'sagemaker/sentiment_rnn'\n",
        "\n",
        "role = sagemaker.get_execution_role()"
    ]
},
{
    "cell_type": "code",
    "execution_count": 89,
    "metadata": {},
    "outputs": [],
    "source": [
        "input_data = sagemaker_session.upload_data(path=data_dir,
bucket=bucket, key_prefix=prefix)"
    ]
},

```

```

{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "**NOTE:** The cell above uploads the entire contents of our data
    directory. This includes the `word_dict.pkl` file. This is fortunate
    as we will need this later on when we create an endpoint that accepts
    an arbitrary review. For now, we will just take note of the fact that
    it resides in the data directory (and so also in the S3 training
    bucket) and that we will need to make sure it gets saved in the model
    directory."
  ]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "## Step 4: Build and Train the PyTorch Model\n",
    "\n",
    "In the XGBoost notebook we discussed what a model is in the
    SageMaker framework. In particular, a model comprises three
    objects\n",
    "\n",
    " - Model Artifacts,\n",
    " - Training Code, and\n",
    " - Inference Code,\n",
    "\n",
    "each of which interact with one another. In the XGBoost example
    we used training and inference code that was provided by Amazon. Here
    we will still be using containers provided by Amazon with the added
    benefit of being able to include our own custom code.\n",
    "\n",
    "We will start by implementing our own neural network in PyTorch
    along with a training script. For the purposes of this project we have
    provided the necessary model object in the `model.py` file, inside of
    the `train` folder. You can see the provided implementation by running
    the cell below."
  ]
},
{
  "cell_type": "code",
  "execution_count": 90,
  "metadata": {},
  "outputs": [
    {
      "name": "stdout",
      "output_type": "stream",
      "text": [
        "\u001b[34mimport\u001b[39;49;00m
        \u001b[04m\u001b[36mtorch\u001b[39;49;00m\u001b[04m\u001b[36m.

```

```

\u001b[39;49;00m\u001b[04m\u001b[36mnn\u001b[39;49;00m
\u001b[34mas\u001b[39;49;00m
\u001b[04m\u001b[36mnn\u001b[39;49;00m\r\n",
    "\r\n",
    "\u001b[34mclass\u001b[39;49;00m
\u001b[04m\u001b[32mLSTMClassifier\u001b[39;49;00m(nn.Module):\r\n",
    "    \u001b[33m\"\"\"\u001b[39;49;00m\r\n",
    "    \u001b[33m    This is the simple RNN model we will be using to
perform Sentiment Analysis.\u001b[39;49;00m\r\n",
    "    \u001b[33m    \"\"\"\u001b[39;49;00m\r\n",
    "\r\n",
    "    \u001b[34mdef\u001b[39;49;00m
\u001b[32m__init__\u001b[39;49;00m(\u001b[36mself\u001b[39;49;00m,
embedding_dim, hidden_dim, vocab_size):\r\n",
    "    \u001b[33m\"\"\"\u001b[39;49;00m\r\n",
    "    \u001b[33m        Initialize the model by settingg up the
various layers.\u001b[39;49;00m\r\n",
    "    \u001b[33m        \"\"\"\u001b[39;49;00m\r\n",
    "    \u001b[36msuper\u001b[39;49;00m(LSTMClassifier,
\u001b[36mself\u001b[39;49;00m).\u001b[32m__init__\u001b[39;49;00m()
\r\n",
    "\r\n",
    "    \u001b[36mself\u001b[39;49;00m.embedding =
nn.Embedding(vocab_size, embedding_dim,
padding_idx=\u001b[34m0\u001b[39;49;00m)\r\n",
    "    \u001b[36mself\u001b[39;49;00m.lstm =
nn.LSTM(embedding_dim, hidden_dim)\r\n",
    "    \u001b[36mself\u001b[39;49;00m.dense =
nn.Linear(in_features=hidden_dim,
out_features=\u001b[34m1\u001b[39;49;00m)\r\n",
    "    \u001b[36mself\u001b[39;49;00m.sig = nn.Sigmoid()\r\n",
    "    \r\n",
    "    \u001b[36mself\u001b[39;49;00m.word_dict =
\u001b[34mNone\u001b[39;49;00m\r\n",
    "\r\n",
    "    \u001b[34mdef\u001b[39;49;00m
\u001b[32mforward\u001b[39;49;00m(\u001b[36mself\u001b[39;49;00m, x):
\r\n",
    "    \u001b[33m\"\"\"\u001b[39;49;00m\r\n",
    "    \u001b[33m        Perform a forward pass of our model on some
input.\u001b[39;49;00m\r\n",
    "    \u001b[33m        \"\"\"\u001b[39;49;00m\r\n",
    "    x = x.t()\r\n",
    "    lengths = x[\u001b[34m0\u001b[39;49;00m,:]\r\n",
    "    reviews = x[\u001b[34m1\u001b[39;49;00m,:]\r\n",
    "    embeds =
\u001b[36mself\u001b[39;49;00m.embedding(reviews)\r\n",
    "    lstm_out, _ =
\u001b[36mself\u001b[39;49;00m.lstm(embeds)\r\n",
    "    out = \u001b[36mself\u001b[39;49;00m.dense(lstm_out)

```

```

\r\n",
    "        out = out[lengths - \u001b[34m1\u001b[39;49;00m,
\u001b[36mrange\u001b[39;49;00m(\u001b[36mlen\u001b[39;49;00m(lengths)
)]\r\n",
    "        \u001b[34mreturn\u001b[39;49;00m
\u001b[36mself\u001b[39;49;00m.sig(out.squeeze())\r\n"
    ]
    },
    "source": [
        "!pygmentize train/model.py"
    ]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "The important takeaway from the implementation provided is that
there are three parameters that we may wish to tweak to improve the
performance of our model. These are the embedding dimension, the
hidden dimension and the size of the vocabulary. We will likely want
to make these parameters configurable in the training script so that
if we wish to modify them we do not need to modify the script itself.
We will see how to do this later on. To start we will write some of
the training code in the notebook so that we can more easily diagnose
any issues that arise.\n",
        "\n",
        "First we will load a small portion of the training data set to
use as a sample. It would be very time consuming to try and train the
model completely in the notebook as we do not have access to a gpu and
the compute instance that we are using is not particularly powerful.
However, we can work on a small bit of the data to get a feel for how
our training script is behaving."
    ]
},
{
    "cell_type": "code",
    "execution_count": 91,
    "metadata": {},
    "outputs": [],
    "source": [
        "import torch\n",
        "import torch.utils.data\n",
        "\n",
        "# Read in only the first 250 rows\n",
        "train_sample = pd.read_csv(os.path.join(data_dir, 'train.csv'),
header=None, names=None, nrows=250)\n",
        "\n",
        "# Turn the input pandas dataframe into tensors\n",
        "train_sample_y =

```

```

torch.from_numpy(train_sample[[0]].values).float().squeeze()\n",
    "train_sample_X = torch.from_numpy(train_sample.drop([0],
axis=1).values).long()\n",
    "\n",
    "# Build the dataset\n",
    "train_sample_ds = torch.utils.data.TensorDataset(train_sample_X,
train_sample_y)\n",
    "# Build the dataloader\n",
    "train_sample_dl = torch.utils.data.DataLoader(train_sample_ds,
batch_size=50)"
    ]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "### (TODO) Writing the training method\n",
        "\n",
        "Next we need to write the training code itself. This should be
very similar to training methods that you have written before to train
PyTorch models. We will leave any difficult aspects such as model
saving / loading and parameter loading until a little later."
    ]
},
{
    "cell_type": "code",
    "execution_count": 92,
    "metadata": {},
    "outputs": [],
    "source": [
        "def train(model, train_loader, epochs, optimizer, loss_fn,
device):\n",
        "    for epoch in range(1, epochs + 1):\n",
        "        model.train()\n",
        "        total_loss = 0\n",
        "        for batch in train_loader:
        "            batch_X, batch_y = batch
        "            \n",
        "            batch_X = batch_X.to(device)\n",
        "            batch_y = batch_y.to(device)\n",
        "            \n",
        "            # TODO: Complete this train method to train the model
provided.\n",
        "            optimizer.zero_grad()\n",
        "            out = model.forward(batch_X)\n",
        "            loss = loss_fn(out, batch_y)\n",
        "            loss.backward()\n",
        "            optimizer.step()\n",
        "            \n",
        "            total_loss += loss.data.item()

```

```

        "        print(\"Epoch: {}, BCELoss: {}\".format(epoch,
total_loss / len(train_loader)))"
    ],
    {
        "cell_type": "markdown",
        "metadata": {},
        "source": [
            "Supposing we have the training method above, we will test that it
is working by writing a bit of code in the notebook that executes our
training method on the small sample training set that we loaded
earlier. The reason for doing this in the notebook is so that we have
an opportunity to fix any errors that arise early when they are easier
to diagnose."
        ]
    },
    {
        "cell_type": "code",
        "execution_count": 93,
        "metadata": {},
        "outputs": [
            {
                "name": "stdout",
                "output_type": "stream",
                "text": [
                    "Epoch: 1, BCELoss: 0.694747531414032\n",
                    "Epoch: 2, BCELoss: 0.6865064024925231\n",
                    "Epoch: 3, BCELoss: 0.6799317836761475\n",
                    "Epoch: 4, BCELoss: 0.6727641105651856\n",
                    "Epoch: 5, BCELoss: 0.6641211748123169\n"
                ]
            }
        ],
        "source": [
            "import torch.optim as optim\n",
            "from train.model import LSTMClassifier\n",
            "\n",
            "device = torch.device(\"cuda\" if torch.cuda.is_available() else
\"cpu\")\n",
            "model = LSTMClassifier(32, 100, 5000).to(device)\n",
            "optimizer = optim.Adam(model.parameters())\n",
            "loss_fn = torch.nn.BCELoss()\n",
            "\n",
            "train(model, train_sample_dl, 5, optimizer, loss_fn, device)"
        ]
    },
    {
        "cell_type": "markdown",
        "metadata": {},
        "source": [

```


"In order to construct a PyTorch model using SageMaker we must provide SageMaker with a training script. We may optionally include a directory which will be copied to the container and from which our training code will be run. When the training container is executed it will check the uploaded directory (if there is one) for a `requirements.txt` file and install any required Python libraries, after which the training script will be run."

```
]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "### (TODO) Training the model\n",
    "\n",
    "When a PyTorch model is constructed in SageMaker, an entry point must be specified. This is the Python file which will be executed when the model is trained. Inside of the `train` directory is a file called `train.py` which has been provided and which contains most of the necessary code to train our model. The only thing that is missing is the implementation of the `train()` method which you wrote earlier in this notebook.\n",
```

```
    "\n",
    "**TODO**: Copy the `train()` method written above and paste it into the `train/train.py` file where required.\n",
```

```
    "\n",
    "The way that SageMaker passes hyperparameters to the training script is by way of arguments. These arguments can then be parsed and used in the training script. To see how this is done take a look at the provided `train/train.py` file."
  ]
},
{
```

```
  "cell_type": "code",
  "execution_count": 94,
  "metadata": {},
  "outputs": [],
  "source": [
    "from sagemaker.pytorch import PyTorch\n",
    "\n",
    "estimator = PyTorch(entry_point=\"train.py\",\n",
    "                    source_dir=\"train\",\n",
    "                    role=role,\n",
    "                    framework_version='0.4.0',\n",
    "                    train_instance_count=1,\n",
    "                    \n",
    "                    train_instance_type='ml.p2.xlarge',\n",
    "                    hyperparameters={\n",
    "                        'epochs': 10,\n",
    "                        'hidden_dim': 200,
```

```

        "        })\n",
        "#'ml.p2.xlarge'"
    ]
},
{
    "cell_type": "code",
    "execution_count": 95,
    "metadata": {},
    "outputs": [
        {
            "name": "stderr",
            "output_type": "stream",
            "text": [
                "'create_image_uri' will be deprecated in favor of
'ImageURIProvider' class in SageMaker Python SDK v2.\n",
                "'s3_input' class will be renamed to 'TrainingInput' in
SageMaker Python SDK v2.\n",
                "'create_image_uri' will be deprecated in favor of
'ImageURIProvider' class in SageMaker Python SDK v2.\n"
            ]
        },
        {
            "name": "stdout",
            "output_type": "stream",
            "text": [
                "2020-12-26 00:15:37 Starting - Starting the training job...\n",
                "2020-12-26 00:15:39 Starting - Launching requested ML
instances.....\n",
                "2020-12-26 00:16:50 Starting - Preparing the instances for
training.....\n",
                "2020-12-26 00:18:07 Downloading - Downloading input data...\n",
                "2020-12-26 00:18:38 Training - Downloading the training
image.....\u001b[34mbash: cannot set terminal process group (-1):
Inappropriate ioctl for device\u001b[0m\n",
                "\u001b[34mbash: no job control in this shell\u001b[0m\n",
                "\u001b[34m2020-12-26 00:19:22,171 sagemaker-containers INFO
Imported framework sagemaker_pytorch_container.training\u001b[0m\n",
                "\u001b[34m2020-12-26 00:19:22,196
sagemaker_pytorch_container.training INFO      Block until all host DNS
lookups succeed.\u001b[0m\n",
                "\u001b[34m2020-12-26 00:19:22,199
sagemaker_pytorch_container.training INFO      Invoking user training
script.\u001b[0m\n",
                "\u001b[34m2020-12-26 00:19:22,457 sagemaker-containers INFO
Module train does not provide a setup.py. \u001b[0m\n",
                "\u001b[34mGenerating setup.py\u001b[0m\n",
                "\u001b[34m2020-12-26 00:19:22,457 sagemaker-containers INFO
Generating setup.cfg\u001b[0m\n",
                "\u001b[34m2020-12-26 00:19:22,457 sagemaker-containers INFO
Generating MANIFEST.in\u001b[0m\n",

```

```

\u001b[34m2020-12-26 00:19:22,458 sagemaker-containers INFO
Installing module with the following command:\u001b[0m\n",
    "\u001b[34m/usr/bin/python -m pip install -U . -r
requirements.txt\u001b[0m\n",
    "\u001b[34mProcessing /opt/ml/code\u001b[0m\n",
    "\u001b[34mCollecting pandas (from -r requirements.txt (line 1))
\n",
    "    Downloading https://files.pythonhosted.org/packages/
74/24/0cdbf8907e1e3bc5a8da03345c23cbcd7044330bb8f73bb12e711a640a00/
pandas-0.24.2-cp35-cp35m-manylinux1_x86_64.whl (10.0MB)\u001b[0m\n",
    "\u001b[34mCollecting numpy (from -r requirements.txt (line 2))
\u001b[0m\n",
    "\u001b[34m    Downloading https://files.pythonhosted.org/
packages/
b5/36/88723426b4ff576809fec7d73594fe17a35c27f8d01f93637637a29ae25b/
numpy-1.18.5-cp35-cp35m-manylinux1_x86_64.whl (19.9MB)\u001b[0m\n",
    "\n",
    "2020-12-26 00:19:21 Training - Training image download
completed. Training in progress.\u001b[34mCollecting nltk (from -r
requirements.txt (line 3))\n",
    "    Downloading https://files.pythonhosted.org/packages/92/75/
ce35194d8e3022203cca0d2f896dbb88689f9b3fce8e9f9cff942913519d/
nltk-3.5.zip (1.4MB)\u001b[0m\n",
    "\u001b[34mCollecting beautifulsoup4 (from -r requirements.txt
(line 4))\n",
    "    Downloading https://files.pythonhosted.org/packages/d1/41/
e6495bd7d3781cee623ce23ea6ac73282a373088fcd0ddc809a047b18eae/
beautifulsoup4-4.9.3-py3-none-any.whl (115kB)\u001b[0m\n",
    "\u001b[34mCollecting html5lib (from -r requirements.txt (line
5))\n",
    "    Downloading https://files.pythonhosted.org/packages/6c/dd/
a834df6482147d48e225a49515aabc28974ad5a4ca3215c18a882565b028/
html5lib-1.1-py2.py3-none-any.whl (112kB)\u001b[0m\n",
    "\u001b[34mRequirement already satisfied, skipping upgrade:
python-dateutil>=2.5.0 in /usr/local/lib/python3.5/dist-packages (from
pandas->-r requirements.txt (line 1)) (2.7.5)\u001b[0m\n",
    "\u001b[34mCollecting pytz>=2011k (from pandas->-r
requirements.txt (line 1))\n",
    "    Downloading https://files.pythonhosted.org/packages/
89/06/2c2d3034b4d6bf22f2a4ae546d16925898658a33b4400cfb7e2c1e2871a3/
pytz-2020.5-py2.py3-none-any.whl (510kB)\u001b[0m\n",
    "\u001b[34mRequirement already satisfied, skipping upgrade:
click in /usr/local/lib/python3.5/dist-packages (from nltk->-r
requirements.txt (line 3)) (7.0)\u001b[0m\n",
    "\u001b[34mCollecting joblib (from nltk->-r requirements.txt
(line 3))\n",
    "    Downloading https://files.pythonhosted.org/packages/28/5c/
cf6a2b65a321c4a209efcdf64c2689efae2cb62661f8f6f4bb28547cf1bf/
joblib-0.14.1-py2.py3-none-any.whl (294kB)\u001b[0m\n",
    "\u001b[34mCollecting regex (from nltk->-r requirements.txt

```

```
(line 3))\n",
    " Downloading https://files.pythonhosted.org/packages/2e/
e4/3447fed9ab29944333f48730ecff4dca92f0868c5b188d6ab2b2078e32c2/
regex-2020.11.13.tar.gz (694kB)\u001b[0m\n",
    "\u001b[34mCollecting tqdm (from nltk->-r requirements.txt (line
3))\n",
    " Downloading https://files.pythonhosted.org/packages/05/bb/
9403e1f30ed060e16835c9b275620ca89191a41ccc2b995b88efbc32dfd9/
tqdm-4.55.0-py2.py3-none-any.whl (68kB)\u001b[0m\n",
    "\u001b[34mCollecting soupsieve>1.2; python_version >=
\"3.0\" (from beautifulsoup4->-r requirements.txt (line 4))\n",
    " Downloading https://files.pythonhosted.org/packages/02/fb/
1c65691a9aeb7bd6ac2aa505b84cb8b49ac29c976411c6ab3659425e045f/
soupsieve-2.1-py3-none-any.whl\u001b[0m\n",
    "\u001b[34mCollecting webencodings (from html5lib->-r
requirements.txt (line 5))\n",
    " Downloading https://files.pythonhosted.org/packages/
f4/24/2a3e3df732393fed8b3ebf2ec078f05546de641fe1b667ee316ec1dcf3b7/
webencodings-0.5.1-py2.py3-none-any.whl\u001b[0m\n",
    "\u001b[34mRequirement already satisfied, skipping upgrade:
six>=1.9 in /usr/local/lib/python3.5/dist-packages (from html5lib->-r
requirements.txt (line 5)) (1.11.0)\u001b[0m\n",
    "\u001b[34mBuilding wheels for collected packages: nltk, train,
regex\n",
    " Running setup.py bdist_wheel for nltk: started\u001b[0m\n",
    "\u001b[34m Running setup.py bdist_wheel for nltk: finished
with status 'done'\n",
    " Stored in directory: /root/.cache/pip/wheels/ae/8c/3f/
b1fe0ba04555b08b57ab52ab7f86023639a526d8bc8d384306\n",
    " Running setup.py bdist_wheel for train: started\n",
    " Running setup.py bdist_wheel for train: finished with status
'done'\n",
    " Stored in directory: /tmp/pip-ephem-wheel-cache-jbwjp0me/
wheels/35/24/16/37574d11bf9bde50616c67372a334f94fa8356bc7164af8ca3\n",
    " Running setup.py bdist_wheel for regex: started\u001b[0m\n",
    "\u001b[34m Running setup.py bdist_wheel for regex: finished
with status 'done'\n",
    " Stored in directory: /root/.cache/pip/wheels/27/f6/66/
a4243e485a0ebc73dc59033ae26c48e82526f77dbfe158ac59\u001b[0m\n",
    "\u001b[34mSuccessfully built nltk train regex\u001b[0m\n",
    "\u001b[34mInstalling collected packages: pytz, numpy, pandas,
joblib, regex, tqdm, nltk, soupsieve, beautifulsoup4, webencodings,
html5lib, train\u001b[0m\n",
    "\u001b[34m Found existing installation: numpy 1.15.4\n",
    " Uninstalling numpy-1.15.4:\n",
    " Successfully uninstalled numpy-1.15.4\u001b[0m\n",
    "\u001b[34mSuccessfully installed beautifulsoup4-4.9.3
html5lib-1.1 joblib-0.14.1 nltk-3.5 numpy-1.18.5 pandas-0.24.2
pytz-2020.5 regex-2020.11.13 soupsieve-2.1 tqdm-4.55.0 train-1.0.0
webencodings-0.5.1\u001b[0m\n",
```

```

    "\u001b[34mYou are using pip version 18.1, however version
20.3.3 is available.\u001b[0m\n",
    "\u001b[34mYou should consider upgrading via the 'pip install --
upgrade pip' command.\u001b[0m\n",
    "\u001b[34m2020-12-26 00:19:45,875 sagemaker-containers INFO
Invoking user script\n",
    "\u001b[0m\n",
    "\u001b[34mTraining Env:\n",
    "\u001b[0m\n",
    "\u001b[34m{\n",
    "    \"num_gpus\": 1,\n",
    "    \"num_cpus\": 4,\n",
    "    \"module_name\": \"train\",\n",
    "    \"hyperparameters\": {\n",
    "        \"hidden_dim\": 200,\n",
    "        \"epochs\": 10\n",
    "    },\n",
    "    \"job_name\": \"sagemaker-
pytorch-2020-12-26-00-15-37-111\",\n",
    "    \"channel_input_dirs\": {\n",
    "        \"training\": \"/opt/ml/input/data/training\"\n",
    "    },\n",
    "    \"input_dir\": \"/opt/ml/input\",\n",
    "    \"input_config_dir\": \"/opt/ml/input/config\",\n",
    "    \"resource_config\": {\n",
    "        \"hosts\": [\n",
    "            \"algo-1\"\n",
    "        ],\n",
    "        \"current_host\": \"algo-1\",\n",
    "        \"network_interface_name\": \"eth0\"\n",
    "    },\n",
    "    \"output_dir\": \"/opt/ml/output\",\n",
    "    \"model_dir\": \"/opt/ml/model\",\n",
    "    \"output_data_dir\": \"/opt/ml/output/data\",\n",
    "    \"log_level\": 20,\n",
    "    \"network_interface_name\": \"eth0\",\n",
    "    \"framework_module\":
\"sagemaker_pytorch_container.training:main\",\n",
    "    \"user_entry_point\": \"train.py\",\n",
    "    \"additional_framework_parameters\": {},\n",
    "    \"hosts\": [\n",
    "        \"algo-1\"\n",
    "    ],\n",
    "    \"current_host\": \"algo-1\",\n",
    "    \"output_intermediate_dir\": \"/opt/ml/output/
intermediate\",\n",
    "    \"module_dir\": \"s3://sagemaker-us-east-1-771062442852/
sagemaker-pytorch-2020-12-26-00-15-37-111/source/sourcedir.tar.gz\",
    "\n",
    "    \"input_data_config\": {\n",

```

```

        \"training\": {\n",
        \"RecordWrapperType\": \"None\", \n",
        \"S3DistributionType\": \"FullyReplicated\", \n",
        \"TrainingInputMode\": \"File\" \n",
        }\n",
    }\u001b[0m\n",
    \"\u001b[34m}\n",
    \"\u001b[0m\n",
    \"\u001b[34mEnvironment variables:\n",
    \"\u001b[0m\n",
    \"\u001b[34mSM_NETWORK_INTERFACE_NAME=eth0\u001b[0m\n",
    \"\u001b[34mPYTHONPATH=/usr/local/bin:/usr/lib/python35.zip:/usr/
lib/python3.5:/usr/lib/python3.5/plat-x86_64-linux-gnu:/usr/lib/
python3.5/lib-dynload:/usr/local/lib/python3.5/dist-packages:/usr/lib/
python3/dist-packages\u001b[0m\n",
    \"\u001b[34mSM_OUTPUT_DIR=/opt/ml/output\u001b[0m\n",
    \"\u001b[34mSM_INPUT_CONFIG_DIR=/opt/ml/input/config\u001b[0m\n",
    \"\u001b[34mSM_NUM_GPUS=1\u001b[0m\n",
    \"\u001b[34mSM_MODEL_DIR=/opt/ml/model\u001b[0m\n",
    \"\u001b[34mSM_USER_ARGS=[\"--epochs\", \"10\", \"--hidden_dim\",
\"200\"]\u001b[0m\n",
    \"\u001b[34mSM_OUTPUT_INTERMEDIATE_DIR=/opt/ml/output/
intermediate\u001b[0m\n",
    \"\u001b[34mSM_HP_HIDDEN_DIM=200\u001b[0m\n",
    \"\u001b[34mSM_HOSTS=[\"algo-1\"]\u001b[0m\n",
    \"\u001b[34mSM_MODULE_NAME=train\u001b[0m\n",
    \"\u001b[34mSM_FRAMEWORK_PARAMS={}\u001b[0m\n",
    \"\u001b[34mSM_INPUT_DATA_CONFIG={\"training\":
{\"RecordWrapperType\": \"None\", \"S3DistributionType\":
\"FullyReplicated\", \"TrainingInputMode\": \"File\"}}\u001b[0m\n",
    \"\u001b[34mSM_HP_EPOCHS=10\u001b[0m\n",
    \"\u001b[34mSM_HPS={\"epochs\": 10, \"hidden_dim\":
200}\u001b[0m\n",
    \"\u001b[34mSM_NUM_CPUS=4\u001b[0m\n",
    \"\u001b[34mSM_OUTPUT_DATA_DIR=/opt/ml/output/data\u001b[0m\n",

    \"\u001b[34mSM_FRAMEWORK_MODULE=sagemaker_pytorch_container.training:ma
in\u001b[0m\n",
    \"\u001b[34mSM_CURRENT_HOST=algo-1\u001b[0m\n",
    \"\u001b[34mSM_USER_ENTRY_POINT=train.py\u001b[0m\n",
    \"\u001b[34mSM_LOG_LEVEL=20\u001b[0m\n",
    \"\u001b[34mSM_MODULE_DIR=s3://sagemaker-us-east-1-771062442852/
sagemaker-pytorch-2020-12-26-00-15-37-111/source/
sourcedir.tar.gz\u001b[0m\n",
    \"\u001b[34mSM_INPUT_DIR=/opt/ml/input\u001b[0m\n",
    \"\u001b[34mSM_CHANNELS=[\"training\"]\u001b[0m\n",
    \"\u001b[34mSM_CHANNEL_TRAINING=/opt/ml/input/data/
training\u001b[0m\n",
    \"\u001b[34mSM_RESOURCE_CONFIG={\"current_host\": \"algo-1\",
\"hosts\": [\"algo-1\"], \"network_interface_name\": \"eth0\"}

```

```

\u001b[0m\n",
    "\u001b[34mSM_TRAINING_ENV={\"additional_framework_parameters\":
{ },\"channel_input_dirs\":{\"training\":\"/opt/ml/input/data/
training\"},\"current_host\":\"algo-1\",\"framework_module\":
\"sagemaker_pytorch_container.training:main\",\"hosts\":[\"algo-1\"],
\"hyperparameters\":{\"epochs\":10,\"hidden_dim\":200},
\"input_config_dir\":\"/opt/ml/input/config\",\"input_data_config\":
{\"training\":{\"RecordWrapperType\":\"None\",\"S3DistributionType\":
\"FullyReplicated\",\"TrainingInputMode\":\"File\"}},\"input_dir\":\"/
opt/ml/input\",\"job_name\":\"sagemaker-
pytorch-2020-12-26-00-15-37-111\",\"log_level\":20,\"model_dir\":\"/
opt/ml/model\",\"module_dir\":\"s3://sagemaker-us-east-1-771062442852/
sagemaker-pytorch-2020-12-26-00-15-37-111/source/sourcedir.tar.gz\",
\"module_name\":\"train\",\"network_interface_name\":\"eth0\",
\"num_cpus\":4,\"num_gpus\":1,\"output_data_dir\":\"/opt/ml/output/
data\",\"output_dir\":\"/opt/ml/output\",\"output_intermediate_dir\":
\"/opt/ml/output/intermediate\",\"resource_config\":{\"current_host\":
\"algo-1\",\"hosts\":[\"algo-1\"],\"network_interface_name\":
\"eth0\"},\"user_entry_point\":\"train.py\"}\n",
    "\u001b[0m\n",
    "\u001b[34mInvoking script with the following command:\n",
    "\u001b[0m\n",
    "\u001b[34m/usr/bin/python -m train --epochs 10 --hidden_dim
200\n",
    "\n",
    "\u001b[0m\n",
    "\u001b[34mUsing device cuda.\u001b[0m\n",
    "\u001b[34mGet train data loader.\u001b[0m\n",
    "\u001b[34mModel loaded with embedding_dim 32, hidden_dim 200,
vocab_size 5000.\u001b[0m\n"
]
},
{
    "name": "stdout",
    "output_type": "stream",
    "text": [
        "\u001b[34mEpoch: 1, BCELoss: 0.6717461396236809\u001b[0m\n",
        "\u001b[34mEpoch: 2, BCELoss: 0.6244163379377249\u001b[0m\n",
        "\u001b[34mEpoch: 3, BCELoss: 0.556365676072179\u001b[0m\n",
        "\u001b[34mEpoch: 4, BCELoss: 0.5079909216384498\u001b[0m\n",
        "\u001b[34mEpoch: 5, BCELoss: 0.449206642350372\u001b[0m\n",
        "\u001b[34mEpoch: 6, BCELoss: 0.39999188087424453\u001b[0m\n",
        "\u001b[34mEpoch: 7, BCELoss: 0.3583047760992634\u001b[0m\n",
        "\u001b[34mEpoch: 8, BCELoss: 0.3345007628810649\u001b[0m\n",
        "\u001b[34mEpoch: 9, BCELoss: 0.31529118880933643\u001b[0m\n",
        "\u001b[34mEpoch: 10, BCELoss: 0.30694439672693913\u001b[0m\n",
        "\u001b[34m2020-12-26 00:22:43,052 sagemaker-containers INFO
Reporting training SUCCESS\u001b[0m\n",
        "\n",
        "2020-12-26 00:22:51 Uploading - Uploading generated training

```

```

model\n",
    "2020-12-26 00:22:51 Completed - Training job completed\n",
    "Training seconds: 284\n",
    "Billable seconds: 284\n"
]
}
],
"source": [
    "estimator.fit({'training': input_data})"
]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "## Step 5: Testing the model\n",
        "\n",
        "As mentioned at the top of this notebook, we will be testing this
        model by first deploying it and then sending the testing data to the
        deployed endpoint. We will do this so that we can make sure that the
        deployed model is working correctly.\n",
        "\n",
        "## Step 6: Deploy the model for testing\n",
        "\n",
        "Now that we have trained our model, we would like to test it to
        see how it performs. Currently our model takes input of the form
        `review_length, review[500]` where `review[500]` is a sequence of
        `500` integers which describe the words present in the review, encoded
        using `word_dict`. Fortunately for us, SageMaker provides built-in
        inference code for models with simple inputs such as this.\n",
        "\n",
        "There is one thing that we need to provide, however, and that is
        a function which loads the saved model. This function must be called
        `model_fn()` and takes as its only parameter a path to the directory
        where the model artifacts are stored. This function must also be
        present in the python file which we specified as the entry point. In
        our case the model loading function has been provided and so no
        changes need to be made.\n",
        "\n",
        "**NOTE**: When the built-in inference code is run it must import
        the `model_fn()` method from the `train.py` file. This is why the
        training code is wrapped in a main guard ( ie, `if __name__ ==
        '__main__':` )\n",
        "\n",
        "Since we don't need to change anything in the code that was
        uploaded during training, we can simply deploy the current model as-
        is.\n",
        "\n",
        "**NOTE**: When deploying a model you are asking SageMaker to
        launch an compute instance that will wait for data to be sent to it.

```


As a result, this compute instance will continue to run until *you* shut it down. This is important to know since the cost of a deployed endpoint depends on how long it has been running for.\n",

"\n",
"In other words **If you are no longer using a deployed endpoint, shut it down!**\n",

"\n",
"**TODO:** Deploy the trained model."
]
},
{
"cell_type": "code",
"execution_count": 99,
"metadata": {},
"outputs": [
{
"name": "stderr",
"output_type": "stream",
"text": [
"Parameter image will be renamed to image_uri in SageMaker
Python SDK v2.\n",
"'create_image_uri' will be deprecated in favor of
'ImageURIProvider' class in SageMaker Python SDK v2.\n",
"Using already existing model: sagemaker-
pytorch-2020-12-26-00-15-37-111\n"
]
},
{
"ename": "ClientError",
"value": "An error occurred (ValidationException) when calling
the CreateEndpoint operation: Cannot create already existing endpoint
\"arn:aws:sagemaker:us-east-1:771062442852:endpoint/sagemaker-
pytorch-2020-12-26-00-15-37-111\".",
"output_type": "error",
"traceback": [

"

"\u001b[0;31m-----
-----\u001b[0m",
"\u001b[0;31mClientError\u001b[0m
Traceback (most recent call last)",
"\u001b[0;32m<ipython-input-99-5bbfacfa3709>\u001b[0m in
\u001b[0;36m<module>\u001b[0;34m\u001b[0m\n\u001b[1;32m1\u001b[0m \u001b[0m \u001b[0;31m# TODO: Deploy the trained
model\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m
001b[0m \u001b[0;32m----> 2\u001b[0m \u001b[0;31m
\u001b[0mpredictor\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m
\u001b[0mestimator\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m
\u001b[0mdeploy\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m
l_instance_count\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m
\u001b[0;36m1\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m

```

\u001b[0minstance_type\u001b[0m \u001b[0;34m=\u001b[0m
\u001b[0;34m'ml.m4.xlarge'\u001b[0m\u001b[0;34m)
\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\n\u001b[0m\u001b[0m",
    "\u001b[0;32m~/anaconda3/envs/pytorch_p36/lib/python3.6/site-
packages/sagemaker/estimator.py\u001b[0m in
\u001b[0;36mdeploy\u001b[0;34m(self, initial_instance_count,
instance_type, accelerator_type, endpoint_name, use_compiled_model,
update_endpoint, wait, model_name, kms_key, data_capture_config, tags,
**kwargs)\u001b[0m\n\u001b[1;32m    743\u001b[0m
\u001b[0mwait\u001b[0m\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0m\u001b[0mwait\u001b[0m\u001b[0m\u001b[0;34m,
1b[0;34m,
\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32m    744\u001b[0m
\u001b[0mkms_key\u001b[0m\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0m\u001b[0mkms_key\u001b[0m\u001b[0m\u001b[0;34m,
0m\u001b[0;34m,
\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\n\u001b[0m\u001b[0;32m--> 745\u001b[0m
\u001b[0mdata_capture_config\u001b[0m\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0m\u001b[0mdata
ta_capture_config\u001b[0m\u001b[0m\u001b[0;34m,
\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32m    746\u001b[0m
)\n\u001b[1;32m    747\u001b[0m
\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0m\u001b[0m",
    "\u001b[0;32m~/anaconda3/envs/pytorch_p36/lib/python3.6/site-
packages/sagemaker/model.py\u001b[0m in
\u001b[0;36mdeploy\u001b[0;34m(self, initial_instance_count,
instance_type, accelerator_type, endpoint_name, update_endpoint, tags,
kms_key, wait, data_capture_config)\u001b[0m\n\u001b[1;32m577\u001b[0m
\u001b[0mkms_key\u001b[0m\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0m\u001b[0mkms_key\u001b[0m\u001b[0m\u001b[0;34m,
0m\u001b[0;34m,
\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32m    578\u001b[0m
\u001b[0mwait\u001b[0m\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0m\u001b[0mwait\u001b[0m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\n\u001b[0m\u001b[0;34m,
1b[0;34m,
\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\n\u001b[0m\u001b[0;32m--> 579\u001b[0m
\u001b[0mdata_capture_config_dict\u001b[0m\u001b[0m\u001b[0;34m=\u001b[0m\u001b[0m\u001b[0mdata
capture_config_dict\u001b[0m\u001b[0m\u001b[0;34m,
\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\n\u001b[0m\u001b[1;32m    580\u001b[0m
)\n\u001b[1;32m    581\u001b[0m
\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0m\u001b[0m",
    "\u001b[0;32m~/anaconda3/envs/pytorch_p36/lib/python3.6/site-
packages/sagemaker/session.py\u001b[0m in
\u001b[0;36mendpoint_from_production_variants\u001b[0;34m(self, name,
production_variants, tags, kms_key, wait, data_capture_config_dict)
\u001b[0m\n\u001b[1;32m    2903\u001b[0m
\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\u001b[0m\n\u001b[1;32m    2904\u001b[0m
\u001b[0mself\u001b[0m\u001b[0m\u001b[0;34m.
\u001b[0msagemaker_client\u001b[0m\u001b[0m\u001b[0;34m.

```

[illegible]

```

\001b[0mkwargs\001b[0m\001b[0;34m)
\001b[0m\001b[0;34m\001b[0m\001b[0;34m\001b[0m\001b[0m\n\001b[0
m\001b[1;32m      358\001b[0m
\001b[0;34m\001b[0m\001b[0m\n\001b[1;32m      359\001b[0m
\001b[0m_api_call\001b[0m\001b[0;34m.
\001b[0m\001b[0m__name__\001b[0m \001b[0;34m=\001b[0m
\001b[0mstr\001b[0m\001b[0;34m(\001b[0m\001b[0m\001b[0m\001b[0m\001b[0m\001b[0m\001b[0m\001b[0m\001b[0;34m)
\001b[0m\001b[0;34m\001b[0m\001b[0;34m\001b[0m\001b[0m\n",
    "\001b[0;32m~/anaconda3/envs/pytorch_p36/lib/python3.6/site-
packages/botocore/client.py\001b[0m in
\001b[0;36m_make_api_call\001b[0;34m(self, operation_name,
api_params)\001b[0m\n\001b[1;32m      674\001b[0m
\001b[0merror_code\001b[0m \001b[0;34m=\001b[0m
\001b[0mparsed_response\001b[0m\001b[0;34m.
\001b[0m\001b[0mget\001b[0m\001b[0;34m(\001b[0m\001b[0;34m"\Erro
r"\001b[0m\001b[0;34m,\001b[0m \001b[0;34m{\001b[0m\001b[0;34m}
\001b[0m\001b[0;34m)\001b[0m\001b[0;34m.
\001b[0m\001b[0mget\001b[0m\001b[0;34m(\001b[0m\001b[0;34m"\Code
"\001b[0m\001b[0;34m)
\001b[0m\001b[0;34m\001b[0m\001b[0;34m\001b[0m\001b[0m\n\001b[1
;32m      675\001b[0m \001b[0merror_class\001b[0m
\001b[0;34m=\001b[0m \001b[0mself\001b[0m\001b[0;34m.
\001b[0m\001b[0mexceptions\001b[0m\001b[0;34m.
\001b[0m\001b[0mfrom_code\001b[0m\001b[0;34m(\001b[0m\001b[0m\001b[0m\001b[0m\001b[0m\001b[0m\001b[0m\001b[0m\001b[0;34m)
\001b[0m\001b[0;34m\001b[0m\001b[0;34m\001b[0m\001b[0m\n\001b[0
;32m--> 676\001b[0;31m \001b[0;32mraise\001b[0m
\001b[0merror_class\001b[0m\001b[0;34m(\001b[0m\001b[0m\001b[0mparsed_res
ponse\001b[0m\001b[0;34m,\001b[0m
\001b[0moperation_name\001b[0m\001b[0;34m)
\001b[0m\001b[0;34m\001b[0m\001b[0;34m\001b[0m\001b[0m\n\001b[0
m\001b[1;32m      677\001b[0m
\001b[0;32melse\001b[0m\001b[0;34m:
\001b[0m\001b[0;34m\001b[0m\001b[0;34m\001b[0m\001b[0m\n\001b[1
;32m      678\001b[0m \001b[0;32mreturn\001b[0m
\001b[0mparsed_response\001b[0m\001b[0;34m\001b[0m\001b[0;34m\00
1b[0m\001b[0m\n",
    "\001b[0;31mClientError\001b[0m: An error occurred
(ValidationException) when calling the CreateEndpoint operation:
Cannot create already existing endpoint \"arn:aws:sagemaker:us-
east-1:771062442852:endpoint/sagemaker-
pytorch-2020-12-26-00-15-37-111\"."
    ]
}
],
"source": [
    "# TODO: Deploy the trained model\n",
    "predictor = estimator.deploy(initial_instance_count = 1,
instance_type = 'ml.m4.xlarge')"
```

```

    ]
  },
  {
    "cell_type": "markdown",
    "metadata": {},
    "source": [
      "## Step 7 - Use the model for testing\n",
      "\n",
      "Once deployed, we can read in the test data and send it off to
our deployed model to get some results. Once we collect all of the
results we can determine how accurate our model is."
    ]
  },
  {
    "cell_type": "code",
    "execution_count": 100,
    "metadata": {},
    "outputs": [],
    "source": [
      "test_X = pd.concat([pd.DataFrame(test_X_len),
pd.DataFrame(test_X)], axis=1)"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": 101,
    "metadata": {},
    "outputs": [],
    "source": [
      "# We split the data into chunks and send each chunk separately,
accumulating the results.\n",
      "\n",
      "def predict(data, rows=512):\n",
      "    split_array = np.array_split(data, int(data.shape[0] /\nfloat(rows) + 1))\n",
      "    predictions = np.array([])\n",
      "    for array in split_array:\n",
      "        predictions = np.append(predictions,\npredictor.predict(array))\n",
      "    \n",
      "    return predictions"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": 102,
    "metadata": {},
    "outputs": [],
    "source": [
      "predictions = predict(test_X.values)\n",

```

```

    "predictions = [round(num) for num in predictions]"
  ]
},
{
  "cell_type": "code",
  "execution_count": 103,
  "metadata": {},
  "outputs": [
    {
      "data": {
        "text/plain": [
          "0.83588"
        ]
      },
      "execution_count": 103,
      "metadata": {},
      "output_type": "execute_result"
    }
  ],
  "source": [
    "from sklearn.metrics import accuracy_score\n",
    "accuracy_score(test_y, predictions)"
  ]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "**Question:** How does this model compare to the XGBoost model  

    you created earlier? Why might these two models perform differently on  

    this dataset? Which do you think is better for sentiment analysis?"
  ]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "**Answer:**\n",
    "\n",
    "This model is better because they take into account the order of  

    words."
  ]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "### (TODO) More testing\n",
    "\n",
    "We now have a trained model which has been deployed and which we

```

can send processed reviews to and which returns the predicted sentiment. However, ultimately we would like to be able to send our model an unprocessed review. That is, we would like to send the review itself as a string. For example, suppose we wish to send the following review to our model."

```
]
},
{
  "cell_type": "code",
  "execution_count": 104,
  "metadata": {},
  "outputs": [],
  "source": [
    "test_review = 'The simplest pleasures in life are the best, and
    this film is one of them. Combining a rather basic storyline of love
    and adventure this movie transcends the usual weekend fair with wit
    and unmitigated charm.'"
  ]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "The question we now need to answer is, how do we send this review
    to our model?\n",
    "\n",
    "Recall in the first section of this notebook we did a bunch of
    data processing to the IMDb dataset. In particular, we did two
    specific things to the provided reviews.\n",
    " - Removed any html tags and stemmed the input\n",
    " - Encoded the review as a sequence of integers using
    `word_dict`\n",
    "\n",
    "In order process the review we will need to repeat these two
    steps.\n",
    "\n",
    "**TODO**: Using the `review_to_words` and `convert_and_pad`
    methods from section one, convert `test_review` into a numpy array
    `test_data` suitable to send to our model. Remember that our model
    expects input of the form `review_length, review[500]`."
  ]
},
{
  "cell_type": "code",
  "execution_count": 105,
  "metadata": {},
  "outputs": [
    {
      "name": "stdout",
      "output_type": "stream",
```

[illegible]

[illegible]

```

    "# TODO: Convert test_review into a form usable by the model and
save the results in test_data\n",
    "test_data = review_to_words(test_review)\n",
    "print('before convert:',test_data)\n",
    "\n",
    "test_data = [np.array(convert_and_pad(word_dict, test_data)[0])]
\n",
    "print('after convert:',test_data)"
    ],
    },
    {
        "cell_type": "markdown",
        "metadata": {},
        "source": [
            "Now that we have processed the review, we can send the resulting
array to our model to predict the sentiment of the review."
        ]
    },
    {
        "cell_type": "code",
        "execution_count": 109,
        "metadata": {},
        "outputs": [
            {
                "data": {
                    "text/plain": [
                        "array(0.5300385, dtype=float32)"
                    ]
                },
                "execution_count": 109,
                "metadata": {},
                "output_type": "execute_result"
            }
        ],
        "source": [
            "predictor.predict(test_data)"
        ]
    },
    {
        "cell_type": "markdown",
        "metadata": {},
        "source": [
            "Since the return value of our model is close to `1`, we can be
certain that the review we submitted is positive."
        ]
    },
    {
        "cell_type": "markdown",
        "metadata": {},
        "source": [

```

```

    """### Delete the endpoint\n",
    "\n",
    "Of course, just like in the XGBoost notebook, once we've deployed
an endpoint it continues to run until we tell it to shut down. Since
we are done using our endpoint for now, we can delete it."
]
},
{
    "cell_type": "code",
    "execution_count": 107,
    "metadata": {},
    "outputs": [
        {
            "name": "stderr",
            "output_type": "stream",
            "text": [
                "estimator.delete_endpoint() will be deprecated in SageMaker
Python SDK v2. Please use the delete_endpoint() function on your
predictor instead.\n"
            ]
        }
    ],
    "source": [
        "estimator.delete_endpoint()"
    ]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "## Step 6 (again) – Deploy the model for the web app\n",
        "\n",
        "Now that we know that our model is working, it's time to create
some custom inference code so that we can send the model a review
which has not been processed and have it determine the sentiment of
the review.\n",
        "\n",
        "As we saw above, by default the estimator which we created, when
deployed, will use the entry script and directory which we provided
when creating the model. However, since we now wish to accept a string
as input and our model expects a processed review, we need to write
some custom inference code.\n",
        "\n",
        "We will store the code that we write in the `serve` directory.
Provided in this directory is the `model.py` file that we used to
construct our model, a `utils.py` file which contains the
`review_to_words` and `convert_and_pad` pre-processing functions which
we used during the initial data processing, and `predict.py`, the file
which will contain our custom inference code. Note also that
`requirements.txt` is present which will tell SageMaker what Python

```

```
libraries are required by our custom inference code.\n",
"\n",
    "When deploying a PyTorch model in SageMaker, you are expected to provide four functions which the SageMaker inference container will use.\n",
    " - `model_fn`: This function is the same function that we used in the training script and it tells SageMaker how to load our model.\n",
    " - `input_fn`: This function receives the raw serialized input that has been sent to the model's endpoint and its job is to de-serialize and make the input available for the inference code.\n",
    " - `output_fn`: This function takes the output of the inference code and its job is to serialize this output and return it to the caller of the model's endpoint.\n",
    " - `predict_fn`: The heart of the inference script, this is where the actual prediction is done and is the function which you will need to complete.\n",
"\n",
    "For the simple website that we are constructing during this project, the `input_fn` and `output_fn` methods are relatively straightforward. We only require being able to accept a string as input and we expect to return a single value as output. You might imagine though that in a more complex application the input or output may be image data or some other binary data which would require some effort to serialize.\n",
"\n",
    "### (TODO) Writing inference code\n",
"\n",
    "Before writing our custom inference code, we will begin by taking a look at the code which has been provided."
]
},
{
    "cell_type": "code",
    "execution_count": 110,
    "metadata": {},
    "outputs": [
        {
            "name": "stdout",
            "output_type": "stream",
            "text": [
                "\u001b[34mimport\u001b[39;49;00m\n\u001b[04m\u001b[36mparse\u001b[39;49;00m\r\n",
                "\u001b[34mimport\u001b[39;49;00m\n\u001b[04m\u001b[36mjson\u001b[39;49;00m\r\n",
                "\u001b[34mimport\u001b[39;49;00m\n\u001b[04m\u001b[36mos\u001b[39;49;00m\r\n",
                "\u001b[34mimport\u001b[39;49;00m\n\u001b[04m\u001b[36mpickle\u001b[39;49;00m\r\n",
                "\u001b[34mimport\u001b[39;49;00m\n\u001b[04m\u001b[36msys\u001b[39;49;00m\r\n",
```

```

        "\u001b[34mimport\u001b[39;49;00m
\u001b[04m\u001b[36msagemaker_containers\u001b[39;49;00m\r\n",
        "\u001b[34mimport\u001b[39;49;00m
\u001b[04m\u001b[36mpandas\u001b[39;49;00m
\u001b[34mas\u001b[39;49;00m
\u001b[04m\u001b[36mpd\u001b[39;49;00m\r\n",
        "\u001b[34mimport\u001b[39;49;00m
\u001b[04m\u001b[36mnumpy\u001b[39;49;00m \u001b[34mas\u001b[39;49;00m
\u001b[04m\u001b[36mnp\u001b[39;49;00m\r\n",
        "\u001b[34mimport\u001b[39;49;00m
\u001b[04m\u001b[36mtorch\u001b[39;49;00m\r\n",
        "\u001b[34mimport\u001b[39;49;00m
\u001b[04m\u001b[36mtorch\u001b[39;49;00m\u001b[04m\u001b[36m.
\u001b[39;49;00m\u001b[04m\u001b[36mnn\u001b[39;49;00m
\u001b[34mas\u001b[39;49;00m
\u001b[04m\u001b[36mnn\u001b[39;49;00m\r\n",
        "\u001b[34mimport\u001b[39;49;00m
\u001b[04m\u001b[36mtorch\u001b[39;49;00m\u001b[04m\u001b[36m.
\u001b[39;49;00m\u001b[04m\u001b[36moptim\u001b[39;49;00m
\u001b[34mas\u001b[39;49;00m
\u001b[04m\u001b[36moptim\u001b[39;49;00m\r\n",
        "\u001b[34mimport\u001b[39;49;00m
\u001b[04m\u001b[36mtorch\u001b[39;49;00m\u001b[04m\u001b[36m.
\u001b[39;49;00m\u001b[04m\u001b[36mutils\u001b[39;49;00m\u001b[04m\u001b[36m.
01b[36m.\u001b[39;49;00m\u001b[04m\u001b[36mdata\u001b[39;49;00m\r\n",
        "\r\n",
        "\u001b[34mfrom\u001b[39;49;00m
\u001b[04m\u001b[36mmodel\u001b[39;49;00m
\u001b[34mimport\u001b[39;49;00m LSTMClassifier\r\n",
        "\r\n",
        "\u001b[34mfrom\u001b[39;49;00m
\u001b[04m\u001b[36mutils\u001b[39;49;00m
\u001b[34mimport\u001b[39;49;00m review_to_words,
convert_and_pad\r\n",
        "\r\n",
        "\u001b[34mdef\u001b[39;49;00m
\u001b[32mmodel_fn\u001b[39;49;00m(model_dir):\r\n",
        "    \u001b[33m\"\"\"Load the PyTorch model from the `model_dir`
directory.\"\"\"\u001b[39;49;00m\r\n",
        "
\u001b[36mprint\u001b[39;49;00m(\u001b[33m\"\"
Loading model.\u001b[39;49;00m\u001b[33m\"\"
\r\n",
        "\r\n",
        "    \u001b[37m# First, load the parameters used to create the
model.\u001b[39;49;00m\r\n",
        "    model_info = {}\r\n",
        "    model_info_path = os.path.join(model_dir,
\u001b[33m'\u001b[39;49;00m\u001b[33mmodel_info.pth\u001b[39;49;00m\u001b[33m'
01b[33m'\u001b[39;49;00m)\r\n",
        "    \u001b[34mwith\u001b[39;49;00m

```

```

\u001b[36mopen\u001b[39;49;00m(model_info_path,
\u001b[33m'\u001b[39;49;00m\u001b[33mrb\u001b[39;49;00m\u001b[33m'\u00
1b[39;49;00m) \u001b[34mas\u001b[39;49;00m f:\r\n",
    "        model_info = torch.load(f)\r\n",
    "\r\n",
    "
\u001b[36mprint\u001b[39;49;00m(\u001b[33m"\u001b[39;49;00m\u001b[33m
model_info: \u001b[39;49;00m\u001b[33m{
\u001b[39;49;00m\u001b[33m"\u001b[39;49;00m.format(model_info))\r\n",
    "\r\n",
    "    \u001b[37m# Determine the device and construct the model.
\u001b[39;49;00m\r\n",
    "    device =
torch.device(\u001b[33m"\u001b[39;49;00m\u001b[33mcuda\u001b[39;49;00
m\u001b[33m"\u001b[39;49;00m \u001b[34mif\u001b[39;49;00m
torch.cuda.is_available() \u001b[34melse\u001b[39;49;00m
\u001b[33m"\u001b[39;49;00m\u001b[33mcpu\u001b[39;49;00m\u001b[33m"\u001b[39;49;00m)\r\n",
    "    model =
LSTMClassifier(model_info[\u001b[33m'\u001b[39;49;00m\u001b[33membeddi
ng_dim\u001b[39;49;00m\u001b[33m'\u001b[39;49;00m],
model_info[\u001b[33m'\u001b[39;49;00m\u001b[33mhidden_dim\u001b[39;49
;00m\u001b[33m'\u001b[39;49;00m],
model_info[\u001b[33m'\u001b[39;49;00m\u001b[33mvocab_size\u001b[39;49
;00m\u001b[33m'\u001b[39;49;00m])\r\n",
    "\r\n",
    "    \u001b[37m# Load the store model parameters.
\u001b[39;49;00m\r\n",
    "    model_path = os.path.join(model_dir,
\u001b[33m'\u001b[39;49;00m\u001b[33mmodel.pth\u001b[39;49;00m\u001b[33m'
3m'\u001b[39;49;00m)\r\n",
    "    \u001b[34mwith\u001b[39;49;00m
\u001b[36mopen\u001b[39;49;00m(model_path,
\u001b[33m'\u001b[39;49;00m\u001b[33mrb\u001b[39;49;00m\u001b[33m'\u00
1b[39;49;00m) \u001b[34mas\u001b[39;49;00m f:\r\n",
    "        model.load_state_dict(torch.load(f))\r\n",
    "\r\n",
    "    \u001b[37m# Load the saved word_dict.\u001b[39;49;00m\r\n",
    "    word_dict_path = os.path.join(model_dir,
\u001b[33m'\u001b[39;49;00m\u001b[33mword_dict.pkl\u001b[39;49;00m\u00
1b[33m'\u001b[39;49;00m)\r\n",
    "    \u001b[34mwith\u001b[39;49;00m
\u001b[36mopen\u001b[39;49;00m(word_dict_path,
\u001b[33m'\u001b[39;49;00m\u001b[33mrb\u001b[39;49;00m\u001b[33m'\u00
1b[39;49;00m) \u001b[34mas\u001b[39;49;00m f:\r\n",
    "        model.word_dict = pickle.load(f)\r\n",
    "\r\n",
    "    model.to(device).eval()\r\n",
    "\r\n",
    "

```

```

\u001b[36mprint\u001b[39;49;00m(\u001b[33m"\u001b[39;49;00m\u001b[33m
Done loading model.\u001b[39;49;00m\u001b[33m"\u001b[39;49;00m)\r\n",
    "\u001b[34mreturn\u001b[39;49;00m model\r\n",
    "\r\n",
    "\u001b[34mdef\u001b[39;49;00m
\u001b[32minput_fn\u001b[39;49;00m(serialized_input_data,
content_type):\r\n",
    "
\u001b[36mprint\u001b[39;49;00m(\u001b[33m'\u001b[39;49;00m\u001b[33mD
eserializing the input data.
\u001b[39;49;00m\u001b[33m'\u001b[39;49;00m)\r\n",
    "\u001b[34mif\u001b[39;49;00m content_type ==
\u001b[33m'\u001b[39;49;00m\u001b[33mtext/
plain\u001b[39;49;00m\u001b[33m'\u001b[39;49;00m:\r\n",
    "        data =
serialized_input_data.decode(\u001b[33m'\u001b[39;49;00m\u001b[33mutf-
8\u001b[39;49;00m\u001b[33m'\u001b[39;49;00m)\r\n",
    "\u001b[34mreturn\u001b[39;49;00m data\r\n",
    "\u001b[34mraise\u001b[39;49;00m
\u001b[36mException\u001b[39;49;00m(\u001b[33m'\u001b[39;49;00m\u001b[33m[
33mRequested unsupported ContentType in content_type:
\u001b[39;49;00m\u001b[33m'\u001b[39;49;00m + content_type)\r\n",
    "\r\n",
    "\u001b[34mdef\u001b[39;49;00m
\u001b[32moutput_fn\u001b[39;49;00m(prediction_output, accept):\r\n",
    "
\u001b[36mprint\u001b[39;49;00m(\u001b[33m'\u001b[39;49;00m\u001b[33mS
erializing the generated output.
\u001b[39;49;00m\u001b[33m'\u001b[39;49;00m)\r\n",
    "\u001b[34mreturn\u001b[39;49;00m
\u001b[36mstr\u001b[39;49;00m(prediction_output)\r\n",
    "\r\n",
    "\u001b[34mdef\u001b[39;49;00m
\u001b[32mpredict_fn\u001b[39;49;00m(input_data, model):\r\n",
    "
\u001b[36mprint\u001b[39;49;00m(\u001b[33m'\u001b[39;49;00m\u001b[33mI
nferring sentiment of input data.
\u001b[39;49;00m\u001b[33m'\u001b[39;49;00m)\r\n",
    "\r\n",
    "        device =
torch.device(\u001b[33m"\u001b[39;49;00m\u001b[33mcuda\u001b[39;49;00
m\u001b[33m"\u001b[39;49;00m \u001b[34mif\u001b[39;49;00m
torch.cuda.is_available() \u001b[34melse\u001b[39;49;00m
\u001b[33m"\u001b[39;49;00m\u001b[33mcpu\u001b[39;49;00m\u001b[33m"\u001b[39;49;00m)\r\n",
    "        \r\n",
    "        \u001b[34mif\u001b[39;49;00m model.word_dict
\u001b[35mis\u001b[39;49;00m \u001b[34mNone\u001b[39;49;00m:\r\n",
    "            \u001b[34mraise\u001b[39;49;00m
\u001b[36mException\u001b[39;49;00m(\u001b[33m'\u001b[39;49;00m\u001b[33m[

```

```

33mModel has not been loaded properly, no word_dict.
\u001b[39;49;00m\u001b[33m'\u001b[39;49;00m)\r\n",
    "\r\n",
    "\u001b[37m# TODO: Process input_data so that it is ready to
be sent to our model.\u001b[39;49;00m\r\n",
    "\u001b[37m#         You should produce two variables:
\u001b[39;49;00m\r\n",
    "\u001b[37m#         data_X    - A sequence of length 500
which represents the converted review\u001b[39;49;00m\r\n",
    "\u001b[37m#         data_len - The length of the
review\u001b[39;49;00m\r\n",
    "\r\n",
    "    data_X ,data_len = convert_and_pad(model.word_dict,
review_to_words(input_data))\r\n",
    "\r\n",
    "\u001b[37m# Using data_X and data_len we construct an
appropriate input tensor. Remember\u001b[39;49;00m\r\n",
    "\u001b[37m# that our model expects input data of the form
'len, review[500]'.\u001b[39;49;00m\r\n",
    "    data_pack = np.hstack((data_len, data_X))\r\n",
    "    data_pack = data_pack.reshape(\u001b[34m1\u001b[39;49;00m,
-\u001b[34m1\u001b[39;49;00m)\r\n",
    "\r\n",
    "    data = torch.from_numpy(data_pack)\r\n",
    "    data = data.to(device)\r\n",
    "\r\n",
    "\u001b[37m# Make sure to put the model into evaluation
mode\u001b[39;49;00m\r\n",
    "    model.eval()\r\n",
    "\r\n",
    "\u001b[37m# TODO: Compute the result of applying the model
to the input data. The variable `result` should\u001b[39;49;00m\r\n",
    "\u001b[37m#         be a numpy array which contains a single
integer which is either 1 or 0\u001b[39;49;00m\r\n",
    "\r\n",
    "    \u001b[34mwith\u001b[39;49;00m torch.no_grad():\r\n",
    "        output = model.forward(data)\r\n",
    "        \r\n",
    "        result = np.round(output.numpy())\r\n",
    "    \r\n",
    "    \u001b[34mreturn\u001b[39;49;00m result\r\n"
]
}
],
"source": [
"!pygmentize serve/predict.py"
]
},
{
"cell_type": "markdown",

```



```

    "metadata": {},
    "source": [
        "As mentioned earlier, the `model_fn` method is the same as the
one provided in the training code and the `input_fn` and `output_fn`
methods are very simple and your task will be to complete the
`predict_fn` method. Make sure that you save the completed file as
`predict.py` in the `serve` directory.\n",
        "\n",
        "**TODO**: Complete the `predict_fn()` method in the `serve/
predict.py` file."
    ]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "### Deploying the model\n",
        "\n",
        "Now that the custom inference code has been written, we will
create and deploy our model. To begin with, we need to construct a new
PyTorchModel object which points to the model artifacts created during
training and also points to the inference code that we wish to use.
Then we can call the deploy method to launch the deployment container.
\n",
        "\n",
        "**NOTE**: The default behaviour for a deployed PyTorch model is
to assume that any input passed to the predictor is a `numpy` array.
In our case we want to send a string so we need to construct a simple
wrapper around the `RealTimePredictor` class to accomodate simple
strings. In a more complicated situation you may want to provide a
serialization object, for example if you wanted to sent image data."
    ]
},
{
    "cell_type": "code",
    "execution_count": 111,
    "metadata": {},
    "outputs": [
        {
            "name": "stderr",
            "output_type": "stream",
            "text": [
                "Parameter image will be renamed to image_uri in SageMaker
Python SDK v2.\n",
                "'create_image_uri' will be deprecated in favor of
'ImageURIProvider' class in SageMaker Python SDK v2.\n"
            ]
        },
        {
            "name": "stdout",

```

```

        "output_type": "stream",
        "text": [
            "-----!"
        ]
    },
    "source": [
        "from sagemaker.predictor import RealTimePredictor\n",
        "from sagemaker.pytorch import PyTorchModel\n",
        "\n",
        "class StringPredictor(RealTimePredictor):\n",
        "    def __init__(self, endpoint_name, sagemaker_session):\n",
        "        super(StringPredictor, self).__init__(endpoint_name,\n",
sagemaker_session, content_type='text/plain')\n",
        "\n",
        "model = PyTorchModel(model_data=estimator.model_data,\n",
        "                        role = role,\n",
        "                        framework_version='0.4.0',\n",
        "                        entry_point='predict.py',\n",
        "                        source_dir='serve',\n",
        "                        predictor_cls=StringPredictor)\n",
        "predictor = model.deploy(initial_instance_count=1,\ninstance_type='ml.m4.xlarge')
    ]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "### Testing the model\n",
        "\n",
        "Now that we have deployed our model with the custom inference\n",
        "code, we should test to see if everything is working. Here we test our\n",
        "model by loading the first `250` positive and negative reviews and\n",
        "send them to the endpoint, then collect the results. The reason for\n",
        "only sending some of the data is that the amount of time it takes for\n",
        "our model to process the input and then perform inference is quite\n",
        "long and so testing the entire data set would be prohibitive."
    ]
},
{
    "cell_type": "code",
    "execution_count": 112,
    "metadata": {},
    "outputs": [],
    "source": [
        "import glob\n",
        "\n",
        "def test_reviews(data_dir='../data/aclImdb', stop=250):\n",
        "    \n",

```

```

    results = []\n",
    ground = []\n",
    \n",
    # We make sure to test both positive and negative reviews
\n",
    for sentiment in ['pos', 'neg']:\n",
    \n",
    path = os.path.join(data_dir, 'test', sentiment, '*.txt')
\n",
    files = glob.glob(path)\n",
    \n",
    files_read = 0\n",
    \n",
    print('Starting ', sentiment, ' files')\n",
    \n",
    # Iterate through the files and send them to the
predictor\n",
    for f in files:\n",
    with open(f) as review:\n",
    # First, we store the ground truth (was the
review positive or negative)\n",
    if sentiment == 'pos':\n",
    ground.append(1)\n",
    else:\n",
    ground.append(0)\n",
    # Read in the review and convert to 'utf-8' for
transmission via HTTP\n",
    review_input = review.read().encode('utf-8')\n",
    # Send the review to the predictor and store the
results\n",
    \n",
    results.append(float(predictor.predict(review_input)))\n",
    \n",
    # Sending reviews to our endpoint one at a time takes
a while so we\n",
    # only send a small number of reviews\n",
    files_read += 1\n",
    if files_read == stop:\n",
    break\n",
    \n",
    return ground, results"
]

```

```

},
{
  "cell_type": "code",
  "execution_count": 113,
  "metadata": {},
  "outputs": [
    {
      "name": "stdout",

```

```

        "output_type": "stream",
        "text": [
            "Starting pos files\n",
            "Starting neg files\n"
        ]
    },
    "source": [
        "ground, results = test_reviews()"
    ],
    {
        "cell_type": "code",
        "execution_count": 114,
        "metadata": {},
        "outputs": [
            {
                "data": {
                    "text/plain": [
                        "0.866"
                    ]
                },
                "execution_count": 114,
                "metadata": {},
                "output_type": "execute_result"
            }
        ],
        "source": [
            "from sklearn.metrics import accuracy_score\n",
            "accuracy_score(ground, results)"
        ]
    },
    {
        "cell_type": "markdown",
        "metadata": {},
        "source": [
            "As an additional test, we can try sending the `test_review` that we looked at earlier."
        ]
    },
    {
        "cell_type": "code",
        "execution_count": 115,
        "metadata": {},
        "outputs": [
            {
                "data": {
                    "text/plain": [
                        "b'1.0'"
                    ]
                }
            }
        ]
    }

```

```

    },
    "execution_count": 115,
    "metadata": {},
    "output_type": "execute_result"
  }
],
"source": [
  "predictor.predict(test_review)"
]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "Now that we know our endpoint is working as expected, we can set
    up the web page that will interact with it. If you don't have time to
    finish the project now, make sure to skip down to the end of this
    notebook and shut down your endpoint. You can deploy it again when you
    come back."
  ]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "## Step 7 (again): Use the model for the web app\n",
    "\n",
    "> **TODO:** This entire section and the next contain tasks for
    you to complete, mostly using the AWS console.\n",
    "\n",
    "So far we have been accessing our model endpoint by constructing
    a predictor object which uses the endpoint and then just using the
    predictor object to perform inference. What if we wanted to create a
    web app which accessed our model? The way things are set up currently
    makes that not possible since in order to access a SageMaker endpoint
    the app would first have to authenticate with AWS using an IAM role
    which included access to SageMaker endpoints. However, there is an
    easier way! We just need to use some additional AWS services.\n",
    "\n",
    "<img src=\"Web App Diagram.svg\">\n",
    "\n",
    "The diagram above gives an overview of how the various services
    will work together. On the far right is the model which we trained
    above and which is deployed using SageMaker. On the far left is our
    web app that collects a user's movie review, sends it off and expects
    a positive or negative sentiment in return.\n",
    "\n",
    "In the middle is where some of the magic happens. We will
    construct a Lambda function, which you can think of as a
    straightforward Python function that can be executed whenever a
  ]
}

```

specified event occurs. We will give this function permission to send and receive data from a SageMaker endpoint.\n",

"\n",

"Lastly, the method we will use to execute the Lambda function is a new endpoint that we will create using API Gateway. This endpoint will be a url that listens for data to be sent to it. Once it gets some data it will pass that data on to the Lambda function and then return whatever the Lambda function returns. Essentially it will act as an interface that lets our web app communicate with the Lambda function.\n",

"\n",

"### Setting up a Lambda function\n",

"\n",

"The first thing we are going to do is set up a Lambda function. This Lambda function will be executed whenever our public API has data sent to it. When it is executed it will receive the data, perform any sort of processing that is required, send the data (the review) to the SageMaker endpoint we've created and then return the result.\n",

"\n",

"#### Part A: Create an IAM Role for the Lambda function\n",

"\n",

"Since we want the Lambda function to call a SageMaker endpoint, we need to make sure that it has permission to do so. To do this, we will construct a role that we can later give the Lambda function.\n",

"\n",

"Using the AWS Console, navigate to the **IAM** page and click on **Roles**. Then, click on **Create role**. Make sure that the **AWS service** is the type of trusted entity selected and choose **Lambda** as the service that will use this role, then click **Next: Permissions**.\n",

"\n",

"In the search box type `sagemaker` and select the check box next to the **AmazonSageMakerFullAccess** policy. Then, click on **Next: Review**.\n",

"\n",

"Lastly, give this role a name. Make sure you use a name that you will remember later on, for example `LambdaSageMakerRole`. Then, click on **Create role**.\n",

"\n",

"#### Part B: Create a Lambda function\n",

"\n",

"Now it is time to actually create the Lambda function.\n",

"\n",

"Using the AWS Console, navigate to the AWS Lambda page and click on **Create a function**. When you get to the next page, make sure that **Author from scratch** is selected. Now, name your Lambda function, using a name that you will remember later on, for example `sentiment_analysis_func`. Make sure that the **Python 3.6** runtime is selected and then choose the role that you created in the previous part. Then, click on **Create Function**.\n",

```

"\n",
"On the next page you will see some information about the Lambda
function you've just created. If you scroll down you should see an
editor in which you can write the code that will be executed when your
Lambda function is triggered. In our example, we will use the code
below. \n",
"\n",
"```python\n",
"# We need to use the low-level library to interact with SageMaker
since the SageMaker API\n",
"# is not available natively through Lambda.\n",
"import boto3\n",
"\n",
"def lambda_handler(event, context):\n",
"\n",
"    # The SageMaker runtime is what allows us to invoke the
endpoint that we've created.\n",
"    runtime = boto3.Session().client('sagemaker-runtime')\n",
"\n",
"    # Now we use the SageMaker runtime to invoke our endpoint,
sending the review we were given\n",
"    response = runtime.invoke_endpoint(EndpointName = '**ENDPOINT
NAME HERE**',      # The name of the endpoint we created\n",
"                                   ContentType = 'text/
plain',              # The data format that is expected\n",
"                                   Body = event['body'])
# The actual review\n",
"\n",
"    # The response is an HTTP response whose body contains the
result of our inference\n",
"    result = response['Body'].read().decode('utf-8')\n",
"\n",
"    return {\n",
"        'statusCode' : 200,\n",
"        'headers' : { 'Content-Type' : 'text/plain', 'Access-
Control-Allow-Origin' : '*' },\n",
"        'body' : result\n",
"    }\n",
"```\n",
"\n",
"Once you have copy and pasted the code above into the Lambda code
editor, replace the `**ENDPOINT NAME HERE**` portion with the name of
the endpoint that we deployed earlier. You can determine the name of
the endpoint using the code cell below."
]
},
{
"cell_type": "code",
"execution_count": 116,
"metadata": {},

```

```

"outputs": [
  {
    "data": {
      "text/plain": [
        "'sagemaker-pytorch-2020-12-26-00-39-01-082'"
      ]
    },
    "execution_count": 116,
    "metadata": {},
    "output_type": "execute_result"
  }
],
"source": [
  "predictor.endpoint"
]
},
{
  "cell_type": "markdown",
  "metadata": {},
  "source": [
    "Once you have added the endpoint name to the Lambda function, click on Save. Your Lambda function is now up and running. Next we need to create a way for our web app to execute the Lambda function.\n",
    "\n",
    "### Setting up API Gateway\n",
    "\n",
    "Now that our Lambda function is set up, it is time to create a new API using API Gateway that will trigger the Lambda function we have just created.\n",
    "\n",
    "Using AWS Console, navigate to Amazon API Gateway and then click on Get started.\n",
    "\n",
    "On the next page, make sure that New API is selected and give the new api a name, for example, `sentiment_analysis_api`. Then, click on Create API.\n",
    "\n",
    "Now we have created an API, however it doesn't currently do anything. What we want it to do is to trigger the Lambda function that we created earlier.\n",
    "\n",
    "Select the Actions dropdown menu and click Create Method. A new blank method will be created, select its dropdown menu and select POST, then click on the check mark beside it.\n",
    "\n",
    "For the integration point, make sure that Lambda Function is selected and click on the Use Lambda Proxy integration. This option makes sure that the data that is sent to the API is then sent directly to the Lambda function with no processing. It also means that
  ]
}

```



```

it down when you don't need it, otherwise you will end up with a
surprisingly large AWS bill.\n",
    "\n",
    "**TODO:** Make sure that you include the edited `index.html` file
in your project submission."
]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "Now that your web app is working, trying playing around with it
and see how well it works.\n",
        "\n",
        "**Question:** Give an example of a review that you entered into
your web app. What was the predicted sentiment of your example
review?"
    ]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "**Answer:**\n",
        "\n",
        "I entered in: \"This movie was great!\"\n",
        "The predicted sentiment is positive."
    ]
},
{
    "cell_type": "markdown",
    "metadata": {},
    "source": [
        "### Delete the endpoint\n",
        "\n",
        "Remember to always shut down your endpoint if you are no longer
using it. You are charged for the length of time that the endpoint is
running so if you forget and leave it on you could end up with an
unexpectedly large bill."
    ]
},
{
    "cell_type": "code",
    "execution_count": null,
    "metadata": {},
    "outputs": [],
    "source": [
        "predictor.delete_endpoint()"
    ]
},
},

```

```

{
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": []
},
{
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": []
}
],
"metadata": {
  "kernelspec": {
    "display_name": "conda_pytorch_p36",
    "language": "python",
    "name": "conda_pytorch_p36"
  },
  "language_info": {
    "codemirror_mode": {
      "name": "ipython",
      "version": 3
    },
    "file_extension": ".py",
    "mimetype": "text/x-python",
    "name": "python",
    "nbconvert_exporter": "python",
    "pygments_lexer": "ipython3",
    "version": "3.6.10"
  }
},
"nbformat": 4,
"nbformat_minor": 2
}

```