Skip-gram Word2Vec

In this notebook, I'll lead you through using PyTorch to implement the [Word2Vec algorithm (https://en.wikipedia.org/wiki/Word2vec)](https://en.wikipedia.org/wiki/Word2vec) using the skip-gram architecture. By implementing this, you'll learn about embedding words for use in natural language processing. This will come in handy when dealing with things like machine translation.

Readings

Here are the resources I used to build this notebook. I suggest reading these either beforehand or while you're working on this material.

- A really good [conceptual overview (http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/)](http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/) of Word2Vec from Chris McCormick
- [First Word2Vec paper (https://arxiv.org/pdf/1301.3781.pdf)](https://arxiv.org/pdf/1301.3781.pdf) from Mikolov et al.
- [Neural Information Processing Systems, paper (http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf)](http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf) with improvements for Word2Vec also from Mikolov et al.

Word embeddings

When you're dealing with words in text, you end up with tens of thousands of word classes to analyze; one for each word in a vocabulary. Trying to one-hot encode these words is massively inefficient because most values in a one-hot vector will be set to zero. So, the matrix multiplication that happens in between a one-hot input vector and a first, hidden layer will result in mostly zero-valued hidden outputs.

To solve this problem and greatly increase the efficiency of our networks, we use what are called **embeddings**. Embeddings are just a fully connected layer like you've seen before. We call this layer the embedding layer and the weights are embedding weights. We skip the multiplication into the embedding layer by instead directly grabbing the hidden layer values from the weight matrix. We can do this because the multiplication of a one-hot encoded vector with a matrix returns the row of the matrix corresponding the index of the "on" input unit.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 8 & 2 & 1 & 9 \\ 6 & 5 & 4 & 0 \\ 7 & 1 & 6 & 2 \\ 1 & 3 & 5 & 8 \\ 0 & 4 & 9 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 5 & 8 \end{bmatrix}$$

Instead of doing the matrix multiplication, we use the weight matrix as a lookup table. We encode the words as integers, for example "heart" is encoded as 958, "mind" as 18094. Then to get hidden layer values for "heart", you just take the 958th row of the embedding matrix. This process is called an **embedding lookup** and the number of hidden units is the **embedding dimension**.

There is nothing magical going on here. The embedding lookup table is just a weight matrix. The embedding layer is just a hidden layer. The lookup is just a shortcut for the matrix multiplication. The lookup table is trained just like any weight matrix.

Embeddings aren't only used for words of course. You can use them for any model where you have a massive number of classes. A particular type of model called **Word2Vec** uses the embedding layer to find vector representations of words that contain semantic meaning.

Word2Vec

The Word2Vec algorithm finds much more efficient representations by finding vectors that represent the words. These vectors also contain semantic information about the words.

I often drink coffee in the mornings.

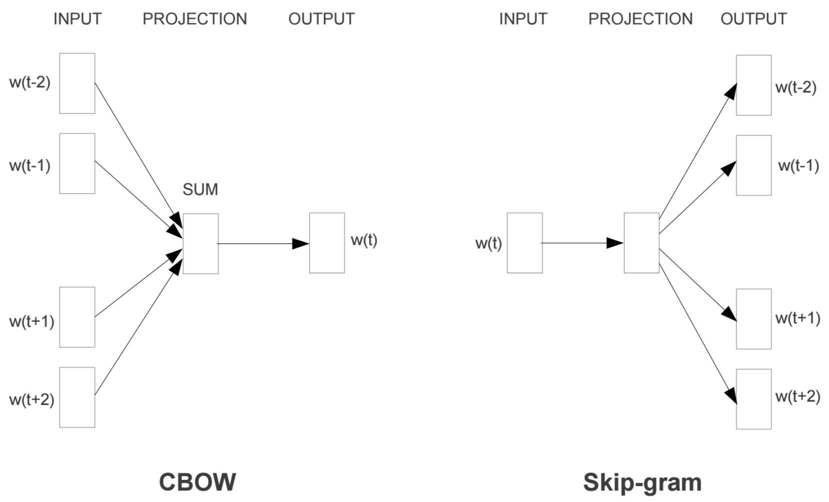
When I'm thirsty, I drink water.

I drink tea, before I go to sleep.

Words that show up in similar **contexts**, such as "coffee", "tea", and "water" will have vectors near each other. Different words will be further away from one another, and relationships can be represented by distance in vector space.

There are two architectures for implementing Word2Vec:

> - CBOW (Continuous Bag-Of-Words) and
> - Skip-gram



In this implementation, we'll be using the **skip-gram architecture** with **negative sampling** because it performs better than CBOW and trains faster with negative sampling. Here, we pass in a word and try to predict the words surrounding it in the text. In this way, we can train the network to learn representations for words that show up in similar contexts.

Loading Data

Next, we'll ask you to load in data and place it in the `data` directory

1. Load the [text8 dataset (https://s3.amazonaws.com/video.udacity-data.com/topher/2018/October/5bbe6499_text8/text8.zip)](https://s3.amazonaws.com/video.udacity-data.com/topher/2018/October/5bbe6499_text8/text8.zip); a file of cleaned up *Wikipedia article text* from Matt Mahoney.
2. Place that data in the `data` folder in the home directory.
3. Then you can extract it and delete the archive, zip file to save storage space.

After following these steps, you should have one file in your data directory: `data/text8` .

In [1]:

```
# read in the extracted text file
with open('data/text8') as f:
    text = f.read()

# print out the first 100 characters
print(text[:100])
```

```
 anarchism originated as a term of abuse first used against early work
ing class radicals including t
```

Pre-processing

Here I'm fixing up the text to make training easier. This comes from the `utils.py` file. The `preprocess` function does a few things:

> - It converts any punctuation into tokens, so a period is changed to `<PERIOD>` . In this data set, there aren't any periods, but it will help in other NLP problems.
> - It removes all words that show up five or *fewer* times in the dataset. This will greatly reduce issues due to noise in the data and improve the quality of the vector representations.
> - It returns a list of words in the text.

This may take a few seconds to run, since our text file is quite large. If you want to write your own functions for this stuff, go for it!

In [2]:

```
import utils

# get list of words
words = utils.preprocess(text)
print(words[:30])
```

```
['anarchism', 'originated', 'as', 'a', 'term', 'of', 'abuse', 'first',
'used', 'against', 'early', 'working', 'class', 'radicals', 'includin
g', 'the', 'diggers', 'of', 'the', 'english', 'revolution', 'and', 'th
e', 'sans', 'culottes', 'of', 'the', 'french', 'revolution', 'whilst']
```

In [3]:

```
# print some stats about this word data
print("Total words in text: {}".format(len(words)))
print("Unique words: {}".format(len(set(words)))) # `set` removes any duplicate wor
```

```
Total words in text: 16680599
Unique words: 63641
```

Dictionaries

Next, I'm creating two dictionaries to convert words to integers and back again (integers to words). This is again done with a function in the `utils.py` file. `create_lookup_tables` takes in a list of words in a text and returns two dictionaries.

> - The integers are assigned in descending frequency order, so the most frequent word ("the") is given the integer 0 and the next most frequent is 1, and so on.

Once we have our dictionaries, the words are converted to integers and stored in the list `int_words` .

In [4]:

```
vocab_to_int, int_to_vocab = utils.create_lookup_tables(words)
int_words = [vocab_to_int[word] for word in words]

print(int_words[:30])
```

```
[5233, 3080, 11, 5, 194, 1, 3133, 45, 58, 155, 127, 741, 476, 10571, 1
33, 0, 27349, 1, 0, 102, 854, 2, 0, 15067, 58112, 1, 0, 150, 854, 358
0]
```

Subsampling

Words that show up often such as "the", "of", and "for" don't provide much context to the nearby words. If we discard some of them, we can remove some of the noise from our data and in return get faster training and better representations. This process is called subsampling by Mikolov. For each word w_i in the training set, we'll discard it with probability given by

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

where t is a threshold parameter and $f(w_i)$ is the frequency of word w_i in the total dataset.

> Implement subsampling for the words in `int_words` . That is, go through `int_words` and discard each word given the probablility $P(w_i)$ shown above. Note that $P(w_i)$ is the probability that a word is discarded. Assign the subsampled data to `train_words` .

In [5]:

```
from collections import Counter
import random
import numpy as np

threshold = 1e-5
word_counts = Counter(int_words)
#print(list(word_counts.items())[0])  # dictionary of int_words, how many times the

total_count = len(int_words)
freqs = {word: count/total_count for word, count in word_counts.items()}
p_drop = {word: 1 - np.sqrt(threshold/freqs[word]) for word in word_counts}
# discard some frequent words, according to the subsampling equation
# create a new list of words for training
train_words = [word for word in int_words if random.random() < (1 - p_drop[word])]

print(train_words[:30])
```

```
[127, 10571, 27349, 15067, 58112, 190, 10712, 104, 2731, 371, 2757, 68
6, 7088, 5233, 320, 44611, 5233, 2621, 8983, 4147, 141, 6437, 4186, 15
3, 5233, 1137, 6, 4860, 6753, 7573]
```

Making batches

Now that our data is in good shape, we need to get it into the proper form to pass it into our network. With the skip-gram architecture, for each word in the text, we want to define a surrounding *context* and grab all the words in a window around that word, with size C.

From [Mikolov et al. (https://arxiv.org/pdf/1301.3781.pdf)](https://arxiv.org/pdf/1301.3781.pdf):

"Since the more distant words are usually less related to the current word than those close to it, we give less weight to the distant words by sampling less from those words in our training examples... If we choose $C = 5$, for each training word we will select randomly a number R in range $[1 : C]$, and then use R words from history and R words from the future of the current word as correct labels."

> **Exercise:** Implement a function `get_target` that receives a list of words, an index, and a window size, then returns a list of words in the window around the index. Make sure to use the algorithm described above, where you chose a random number of words to from the window.

Say, we have an input and we're interested in the idx=2 token, `741`:

```
[5233, 58, 741, 10571, 27349, 0, 15067, 58112, 3580, 58, 10712]
```

For `R=2`, `get_target` should return a list of four values:

```
[5233, 58, 10571, 27349]
```

In [6]:

```python
def get_target(words, idx, window_size=5):
    ''' Get a list of words in a window around an index. '''

    R = np.random.randint(1, window_size+1)
    start = idx - R if (idx - R) > 0 else 0
    stop = idx + R
    target_words = words[start:idx] + words[idx+1:stop+1]

    return list(target_words)
```

In [7]:

```python
# test your code!

# run this cell multiple times to check for random window selection
int_text = [i for i in range(10)]
print('Input: ', int_text)
idx=5 # word index of interest

target = get_target(int_text, idx=idx, window_size=5)
print('Target: ', target)  # you should get some indices around the idx
```

```
Input:  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Target:  [3, 4, 6, 7]
```

Generating Batches

Here's a generator function that returns batches of input and target data for our model, using the `get_target` function from above. The idea is that it grabs `batch_size` words from a words list. Then for each of those batches, it gets the target words in a window.

In [8]:

```
def get_batches(words, batch_size, window_size=5):
    ''' Create a generator of word batches as a tuple (inputs, targets) '''

    n_batches = len(words)//batch_size

    # only full batches
    words = words[:n_batches*batch_size]

    for idx in range(0, len(words), batch_size):
        x, y = [], []
        batch = words[idx:idx+batch_size]
        for ii in range(len(batch)):
            batch_x = batch[ii]
            batch_y = get_target(batch, ii, window_size)
            y.extend(batch_y)
            x.extend([batch_x]*len(batch_y))
        yield x, y
```

In [9]:

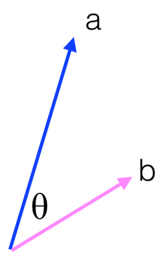
```
int_text = [i for i in range(20)]
x,y = next(get_batches(int_text, batch_size=4, window_size=5))

print('x\n', x)
print('y\n', y)
```

```
x
 [0, 1, 1, 1, 2, 2, 2, 3, 3, 3]
y
 [1, 0, 2, 3, 0, 1, 3, 0, 1, 2]
```

Validation

Here, I'm creating a function that will help us observe our model as it learns. We're going to choose a few common words and few uncommon words. Then, we'll print out the closest words to them using the cosine similarity:



$$\text{similarity} = \cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}$$

We can encode the validation words as vectors $\vec{a}$ using the embedding table, then calculate the similarity with each word vector $\vec{b}$ in the embedding table. With the similarities, we can print out the validation words and words in our embedding table semantically similar to those words. It's a nice way to check that our embedding table is grouping together words with similar semantic meanings.

In [10]:

```
def cosine_similarity(embedding, valid_size=16, valid_window=100, device='cpu'):
    """ Returns the cosine similarity of validation words with words in the embeddi
        Here, embedding should be a PyTorch embedding module.
    """

    # Here we're calculating the cosine similarity between some random words and
    # our embedding vectors. With the similarities, we can look at what words are
    # close to our random words.

    # sim = (a . b) / |a||b|

    embed_vectors = embedding.weight

    # magnitude of embedding vectors, |b|
    magnitudes = embed_vectors.pow(2).sum(dim=1).sqrt().unsqueeze(0)

    # pick N words from our ranges (0,window) and (1000,1000+window). lower id impl
    valid_examples = np.array(random.sample(range(valid_window), valid_size//2))
    valid_examples = np.append(valid_examples,
                               random.sample(range(1000,1000+valid_window), valid_s
    valid_examples = torch.LongTensor(valid_examples).to(device)

    valid_vectors = embedding(valid_examples)
    similarities = torch.mm(valid_vectors, embed_vectors.t())/magnitudes

    return valid_examples, similarities
```

SkipGram model

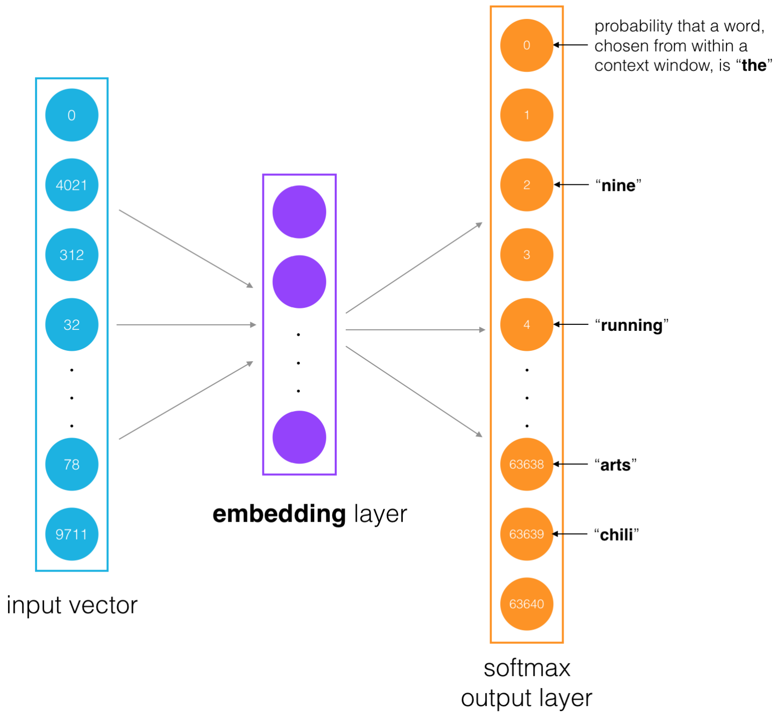
Define and train the SkipGram model.

> You'll need to define an [embedding layer (https://pytorch.org/docs/stable/nn.html#embedding)](https://pytorch.org/docs/stable/nn.html#embedding) and a final, softmax output layer.

An Embedding layer takes in a number of inputs, importantly:

- **num_embeddings** – the size of the dictionary of embeddings, or how many rows you'll want in the embedding weight matrix
- **embedding_dim** – the size of each embedding vector; the embedding dimension

Below is an approximate diagram of the general structure of our network.



> - The input words are passed in as batches of input word tokens.
> - This will go into a hidden layer of linear units (our embedding layer).
> - Then, finally into a softmax output layer.

We'll use the softmax layer to make a prediction about the context words by sampling, as usual.

Negative Sampling

For every example we give the network, we train it using the output from the softmax layer. That means for each input, we're making very small changes to millions of weights even though we only have one true example. This makes training the network very inefficient. We can approximate the loss from the softmax layer by only updating a small subset of all the weights at once. We'll update the weights for the correct example, but only a small number of incorrect, or noise, examples. This is called ["negative sampling" (http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf)](http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf).

There are two modifications we need to make. First, since we're not taking the softmax output over all the words, we're really only concerned with one output word at a time. Similar to how we use an embedding table to map the input word to the hidden layer, we can now use another embedding table to map the hidden layer to the output word. Now we have two embedding layers, one for input words and one for output words. Secondly, we use a modified loss function where we only care about the true example and a small subset of noise examples.

$$-\log \sigma\left(u_{w_O}{}^\top v_{w_I}\right) - \sum_i^N \mathbb{E}_{w_i\sim P_n(w)} \log \sigma\left(-u_{w_i}{}^\top v_{w_I}\right)$$

This is a little complicated so I'll go through it bit by bit. $u_{w_O}{}^\top$ is the embedding vector for our "output" target word (transposed, that's the $^\top$ symbol) and v_{w_I} is the embedding vector for the "input" word. Then the first term

$$\log \sigma \left(u_{w_O}{}^\top v_{w_I} \right)$$

says we take the log-sigmoid of the inner product of the output word vector and the input word vector. Now the second term, let's first look at

$$\sum_i^N \mathbb{E}_{w_i \sim P_n(w)}$$

This means we're going to take a sum over words w_i drawn from a noise distribution $w_i \sim P_n(w)$. The noise distribution is basically our vocabulary of words that aren't in the context of our input word. In effect, we can randomly sample words from our vocabulary to get these words. $P_n(w)$ is an arbitrary probability distribution though, which means we get to decide how to weight the words that we're sampling. This could be a uniform distribution, where we sample all words with equal probability. Or it could be according to the frequency that each word shows up in our text corpus, the unigram distribution $U(w)$. The authors found the best distribution to be $U(w)^{3/4}$, empirically.

Finally, in

$$\log \sigma \left(-u_{w_i}{}^\top v_{w_I} \right),$$

we take the log-sigmoid of the negated inner product of a noise vector with the input vector.

$$-\log \underset{\uparrow \text{correct target}}{\boxed{\sigma \left(u_{w_O}{}^\top v_{w_I} \right)}} - \sum_i^N \mathbb{E}_{w_i \sim P_n(w)} \log \underset{\downarrow \text{noisy target}}{\boxed{\sigma \left(-u_{w_i}{}^\top v_{w_I} \right)}}$$

To give you an intuition for what we're doing here, remember that the sigmoid function returns a probability between 0 and 1. The first term in the loss pushes the probability that our network will predict the correct word w_O towards 1. In the second term, since we are negating the sigmoid input, we're pushing the probabilities of the noise words towards 0.

In [11]:

```
import torch
from torch import nn
import torch.optim as optim
```

In [12]:

```
class SkipGramNeg(nn.Module):
    def __init__(self, n_vocab, n_embed, noise_dist=None):
        super().__init__()

        self.n_vocab = n_vocab
        self.n_embed = n_embed
        self.noise_dist = noise_dist

        # define embedding layers for input and output words
        self.in_embed = nn.Embedding(n_vocab, n_embed)
        self.out_embed = nn.Embedding(n_vocab, n_embed)

        # Initialize embedding tables with uniform distribution
        # I believe this helps with convergence
        self.in_embed.weight.data.uniform_(-1, 1)
        self.out_embed.weight.data.uniform_(-1, 1)

    def forward_input(self, input_words):
        input_vectors = self.in_embed(input_words)
        return input_vectors

    def forward_output(self, output_words):
        output_vectors = self.out_embed(output_words)
        return output_vectors

    def forward_noise(self, batch_size, n_samples):
        """ Generate noise vectors with shape (batch_size, n_samples, n_embed)"""
        if self.noise_dist is None:
            # Sample words uniformly
            noise_dist = torch.ones(self.n_vocab)
        else:
            noise_dist = self.noise_dist

        # Sample words from our noise distribution
        noise_words = torch.multinomial(noise_dist,
                                        batch_size * n_samples,
                                        replacement=True)

        device = "cuda" if model.out_embed.weight.is_cuda else "cpu"
        noise_words = noise_words.to(device)

        noise_vectors = self.out_embed(noise_words).view(batch_size, n_samples, sel

        return noise_vectors
```

In [13]:

```
class NegativeSamplingLoss(nn.Module):
    def __init__(self):
        super().__init__()

    def forward(self, input_vectors, output_vectors, noise_vectors):

        batch_size, embed_size = input_vectors.shape

        # Input vectors should be a batch of column vectors
        input_vectors = input_vectors.view(batch_size, embed_size, 1)

        # Output vectors should be a batch of row vectors
        output_vectors = output_vectors.view(batch_size, 1, embed_size)

        # bmm = batch matrix multiplication
        # correct log-sigmoid loss
        out_loss = torch.bmm(output_vectors, input_vectors).sigmoid().log()
        out_loss = out_loss.squeeze()

        # incorrect log-sigmoid loss
        noise_loss = torch.bmm(noise_vectors.neg(), input_vectors).sigmoid().log()
        noise_loss = noise_loss.squeeze().sum(1)  # sum the losses over the sample

        # negate and sum correct and noisy log-sigmoid losses
        # return average batch loss
        return -(out_loss + noise_loss).mean()
```

Training

Below is our training loop, and I recommend that you train on GPU, if available.

In [14]:

```
device = 'cuda' if torch.cuda.is_available() else 'cpu'

# Get our noise distribution
# Using word frequencies calculated earlier in the notebook
word_freqs = np.array(sorted(freqs.values(), reverse=True))
unigram_dist = word_freqs/word_freqs.sum()
noise_dist = torch.from_numpy(unigram_dist**(0.75)/np.sum(unigram_dist**(0.75)))

# instantiating the model
embedding_dim = 300
model = SkipGramNeg(len(vocab_to_int), embedding_dim, noise_dist=noise_dist).to(dev

# using the loss that we defined
criterion = NegativeSamplingLoss()
optimizer = optim.Adam(model.parameters(), lr=0.003)

print_every = 1500
steps = 0
epochs = 5

# train for some number of epochs
for e in range(epochs):

    # get our input, target batches
    for input_words, target_words in get_batches(train_words, 512):
        steps += 1
        inputs, targets = torch.LongTensor(input_words), torch.LongTensor(target_wo
        inputs, targets = inputs.to(device), targets.to(device)

        # input, output, and noise vectors
        input_vectors = model.forward_input(inputs)
        output_vectors = model.forward_output(targets)
        noise_vectors = model.forward_noise(inputs.shape[0], 5)

        # negative sampling loss
        loss = criterion(input_vectors, output_vectors, noise_vectors)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        # loss stats
        if steps % print_every == 0:
            print("Epoch: {}/{}".format(e+1, epochs))
            print("Loss: ", loss.item()) # avg batch loss at this point in training
            valid_examples, valid_similarities = cosine_similarity(model.in_embed,
            _, closest_idxs = valid_similarities.topk(6)

            valid_examples, closest_idxs = valid_examples.to('cpu'), closest_idxs.t
            for ii, valid_idx in enumerate(valid_examples):
                closest_words = [int_to_vocab[idx.item()] for idx in closest_idxs[i
                print(int_to_vocab[valid_idx.item()] + " | " + ', '.join(closest_wo
            print("...\n")
```

```
Epoch: 1/5
Loss:  6.7322540283203125
to | the, earth, barium, protection, and
zero | the, and, in, of, one
been | pathogen, competing, restored, advertising, demanded
```

```
between | herman, junta, kenyatta, two, outsiders
also | beowulf, creatures, recommendation, cooler, boundary
it | the, that, confer, send, as
a | make, of, harper, that, the
all | studio, blubber, drawbacks, accidentally, autonomous
quite | neighbours, once, casinos, stuttgart, monster
instance | marcus, teeth, leon, peninsular, married
report | in, festivals, member, backbone, strains
pressure | khan, muni, bipolar, swimming, satellite
know | unify, had, klara, shapeshifting, households
stage | clark, mayor, yates, supplements, doubtful
engineering | sacs, z, one, heaven, passes
animals | concentration, flicks, silvertown, absorb, geologically
...

Epoch: 1/5
Loss:  4.835074424743652
his | in, the, a, of, as
often | procurator, of, that, networks, established
had | one, nine, which, zero, in
many | of, which, rocks, with, two
see | of, from, as, with, a
and | the, a, to, of, in
between | two, is, junta, neoliberalism, three
people | with, in, for, alex, some
construction | dutch, universiti, to, testament, atmospheres
award | waqf, era, highway, by, bonding
ocean | malthus, ubiquitous, regimens, who, transiting
engine | unproductive, latina, zoe, by, battlefield
quite | casinos, neighbours, once, portable, monster
pope | comparison, coup, cheek, narrowly, mph
event | soils, cdots, contingent, charlton, endless
lived | disadvantage, saltwater, underwater, overview, donkeys
...

Epoch: 1/5
Loss:  4.341375350952148
when | to, for, by, an, they
or | is, be, a, the, in
only | by, in, to, be, as
known | a, and, nine, were, one
not | to, be, is, that, it
about | to, and, is, of, his
are | is, that, a, in, the
in | the, and, of, a, to
rise | fax, laws, hells, comedienne, colonize
magazine | liberal, delirium, pivot, narnia, taboos
ocean | malthus, zeppelins, size, accuses, ubiquitous
dr | cleopatra, chojn, suburb, aviator, runways
experience | thought, god, affluence, interdiction, an
construction | dutch, pun, of, atmospheres, universiti
instance | marcus, nevertheless, gash, conflict, festivals
bible | babylonian, vibrates, fielded, cbt, guilbert
...

Epoch: 1/5
Loss:  3.6369738578796387
states | united, in, government, seven, state
his | he, the, in, during, with
many | most, by, of, the, from
two | four, five, seven, zero, one
```

```
about | of, the, and, in, zero
war | had, united, the, one, of
they | the, are, a, of, that
eight | nine, one, seven, four, five
police | elections, states, administration, economic, countries
recorded | patass, toothed, mats, reuptake, d
pressure | temperature, use, muni, where, can
road | towns, volvo, gauge, hen, shakespearean
http | www, software, selecting, links, large
egypt | west, israel, european, hobbit, afghanistan
alternative | wikis, imbedded, davenport, slowly, mingus
institute | chocolates, summations, incurring, brushed, christianity
...

Epoch: 1/5
Loss:  3.504284381866455
is | a, the, such, to, of
after | and, nine, one, was, the
have | the, are, from, most, that
nine | one, eight, four, seven, six
may | of, with, their, the, and
time | a, many, in, by, most
see | of, by, are, and, is
with | a, the, in, of, an
creation | yet, god, world, since, thorns
cost | systems, rate, blackened, expensive, taint
something | knowledge, they, follow, question, comments
existence | human, our, argument, teachings, principle
smith | charles, american, william, richard, politician
governor | charles, president, general, queen, executive
police | united, courts, military, prisoners, u
joseph | writer, ii, french, born, d
...

Epoch: 1/5
Loss:  3.3184025287628174
some | are, that, is, still, to
more | these, are, which, to, or
he | his, him, her, after, wife
other | are, and, or, for, is
th | century, history, west, five, roman
however | to, the, which, some, that
such | is, different, or, are, be
would | to, was, they, him, did
http | www, links, com, web, website
creation | pushcha, since, they, to, beginning
instance | can, not, functions, complex, value
something | that, you, your, could, anything
bible | hebrew, testament, christian, jewish, biblical
bill | american, film, nine, george, singer
file | format, user, files, software, computer
quite | have, different, are, such, can
...

Epoch: 2/5
Loss:  2.8340201377868652
six | four, eight, seven, one, three
can | be, or, techniques, types, using
between | and, from, part, thus, the
have | some, but, the, in, are
war | military, forces, troops, killed, soviet
```

```
from | into, and, which, the, in
when | to, the, can, put, them
history | links, see, external, list, and
brother | wife, died, father, his, son
mathematics | mathematical, theory, theories, science, theorem
consists | are, or, note, and, terms
event | football, ever, park, saw, ball
experience | towards, effects, that, what, belief
police | campaign, convicted, military, armed, party
versions | classic, video, features, version, software
egypt | dynasty, persian, east, empire, egyptian
...

Epoch: 2/5
Loss:  2.5524561405181885
were | with, became, heavy, was, in
system | systems, code, computer, programs, using
other | or, these, make, see, often
over | a, and, has, there, or
but | and, that, however, an, of
was | had, after, victory, during, defeat
states | united, military, u, state, armed
during | was, in, first, the, s
except | divided, or, and, some, is
writers | works, fiction, literary, poets, published
something | might, could, our, without, more
road | city, street, park, east, washington
governor | president, appointed, elected, cabinet, minister
engine | engines, speed, models, cylinder, car
san | university, california, los, college, santa
frac | x, equation, f, mathbf, cdot
...

Epoch: 2/5
Loss:  2.7377986907958984
its | the, between, and, more, their
was | he, had, daughter, days, of
it | thus, be, can, into, a
into | to, it, a, the, by
new | university, city, york, history, college
up | and, the, can, will, move
or | non, to, are, a, these
seven | one, eight, nine, four, zero
engineering | engineers, systems, technology, electrical, electronics
marriage | her, married, divorce, life, he
pope | roman, papal, rome, leo, catholics
shown | complex, finite, number, axis, above
egypt | syria, persian, egyptian, cairo, sinai
magazine | publishing, books, bbc, interview, career
grand | des, paris, royal, north, st
channel | satellite, radio, cable, coverage, broadcast
...

Epoch: 2/5
Loss:  2.6995224952697754
no | import, done, jargon, info, duplicate
be | are, any, time, for, these
four | five, three, two, six, nine
five | four, six, three, zero, two
their | and, the, themselves, to, or
system | systems, data, programming, code, implementation
```

```
the | in, of, by, and, to
known | the, from, a, north, in
square | kilometers, situated, located, near, county
versions | computer, macintosh, version, feature, microsoft
magazine | media, newspapers, fiction, bbc, publishing
woman | her, she, husband, mother, birth
shows | television, tv, genre, game, final
assembly | elected, elections, legislative, cabinet, president
bill | office, president, executive, bush, tony
troops | army, military, war, armies, soviet
...

Epoch: 2/5
Loss:  2.580954074859619
to | the, and, when, a, as
th | nd, century, one, zero, four
has | are, other, or, as, include
with | in, of, a, the, also
zero | five, three, four, two, nine
however | in, their, to, have, was
if | we, will, must, so, be
more | are, most, as, or, and
dr | ed, four, story, bibliography, john
recorded | album, recording, song, label, records
engine | engines, cylinder, combustion, fuel, prototype
rise | revolution, influence, thought, attempts, and
award | awards, winners, best, awarded, fame
issue | opinion, court, rejected, amendment, supreme
pressure | heat, gas, effective, forcing, air
versions | version, pc, macintosh, microsoft, os
...

Epoch: 2/5
Loss:  2.7661571502685547
a | which, and, to, the, or
it | to, is, for, an, of
for | and, with, it, a, which
years | year, birth, months, death, female
would | he, wanted, to, him, was
however | to, some, during, it, occur
five | one, six, two, eight, seven
there | are, which, except, typically, in
ice | winter, hockey, temperature, temperatures, lake
pre | main, modern, see, millennium, early
behind | left, told, hitting, driving, doors
existence | defined, follows, interpretation, quantum, notion
governor | appointed, prime, minister, john, elected
scale | scales, temperature, flat, major, instruments
account | potential, hypothesis, how, this, values
freedom | political, accused, presidential, rights, social
...

Epoch: 3/5
Loss:  2.850525379180908
about | people, what, part, according, life
zero | two, five, four, seven, eight
more | are, than, a, its, the
i | we, you, him, me, daughter
up | down, their, to, the, from
to | a, the, for, in, this
d | b, laureate, politician, writer, actor
```

```
were | their, had, forced, took, to
something | we, you, want, sufficiently, feels
marriage | married, marriages, marry, divorce, wife
lived | died, throne, sons, his, mother
pressure | temperature, heat, liquid, melting, pressures
question | questions, ought, belief, argued, ethical
channel | channels, cable, television, tv, radio
report | reports, review, news, pdf, terrorist
mainly | largest, ethnic, large, many, among
...

Epoch: 3/5
Loss:  2.4977896213531494
there | are, is, every, least, any
which | a, is, it, to, be
would | to, him, that, their, lose
between | this, into, non, formed, characterized
can | a, be, is, if, cannot
with | the, to, or, a, some
were | was, had, before, in, who
who | his, married, he, friend, had
liberal | liberals, party, liberalism, conservative, democracy
engineering | technology, engineers, systems, university, education
rise | massive, increasing, decline, wealth, depends
prince | duke, son, princess, sir, elizabeth
troops | army, forces, war, germans, armies
san | francisco, diego, california, jose, santa
dr | robert, lynn, eric, journal, musician
discovered | discovery, found, observations, periodic, obtained
...

Epoch: 3/5
Loss:  2.225531578063965
or | are, is, to, all, other
to | be, the, it, or, able
seven | eight, nine, one, four, three
have | be, were, and, to, some
s | his, one, two, eight, nine
world | most, the, cultural, war, race
called | is, use, are, an, used
where | n, t, to, left, above
gold | silver, copper, metals, tin, diamonds
frac | x, equation, cdot, mathbf, cos
engine | engines, cylinder, fuel, piston, powered
writers | philosophers, poets, novelists, deaths, literature
numerous | in, regarding, other, many, led
report | news, reports, links, cia, org
joseph | john, william, born, benjamin, james
ice | rock, frozen, hockey, snow, crush
...

Epoch: 3/5
Loss:  2.6488029956817627
b | d, k, r, n, y
will | if, would, can, not, should
from | the, of, in, and, with
as | a, such, the, or, in
known | and, the, of, was, roman
he | his, him, himself, her, who
an | which, the, it, on, is
can | used, are, it, be, if
```

```
freedom | social, rights, policies, political, promoting
gold | silver, copper, metals, precious, timber
issue | political, accepted, rejected, commentators, membership
discovered | discovery, observations, experiments, astronomer, found
nobel | prize, laureate, physicist, chemist, politician
governor | secretary, president, appointed, appoints, senate
ice | rock, crust, dry, frozen, rocks
dr | starring, michael, ed, allen, his
...

Epoch: 3/5
Loss:  2.487758159637451
between | there, eastern, in, or, as
years | zero, female, age, three, birth
their | they, and, however, because, only
may | are, usually, means, in, or
over | the, under, zero, and, in
known | th, by, the, also, in
to | and, the, of, in, was
at | in, was, england, near, he
troops | forces, army, invasion, attack, invaded
dr | nine, four, allen, poet, bibliography
numerous | also, around, most, cited, and
professional | football, fame, american, school, canadian
construction | built, constructed, buildings, architectural, building
scale | scales, tend, materials, technological, produce
brother | son, daughter, cousin, sister, wife
ice | hockey, rock, skating, crust, frozen
...

Epoch: 3/5
Loss:  2.500605583190918
have | their, are, this, were, there
their | have, they, some, or, are
use | such, these, applications, possible, used
six | one, two, eight, five, zero
but | it, or, be, the, that
his | he, father, him, brother, son
years | year, death, five, female, age
they | their, to, however, so, would
active | organized, radical, passive, bahasa, member
mathematics | mathematical, mathematicians, euclid, sciences, math
mean | n, mu, argument, variable, sin
running | run, platform, address, runs, on
alternative | music, for, slang, see, used
channel | channels, cable, radio, broadcast, television
shown | in, both, called, here, is
ice | hockey, skating, glacier, frozen, winter
...

Epoch: 4/5
Loss:  2.3018791675567627
more | are, but, only, is, very
its | the, from, of, is, has
first | was, the, for, one, had
which | is, it, in, the, also
this | it, that, in, most, to
th | century, nd, centuries, eight, period
where | in, when, or, to, long
three | two, one, four, zero, six
orthodox | catholic, church, christians, christianity, orthodoxy
```

```
award | awards, nomination, oscar, awarded, best
bbc | weekly, august, listing, broadcasts, news
accepted | believe, according, opinion, interpretation, questions
recorded | album, song, songs, albums, date
square | density, miles, length, mi, km
troops | army, war, forces, armies, battle
professional | football, team, baseball, teams, players
...

Epoch: 4/5
Loss:  2.524277687072754
of | in, and, the, by, s
united | states, british, department, canada, countries
in | of, and, the, was, from
first | one, the, s, was, by
his | he, him, himself, who, father
see | of, is, history, in, links
state | california, university, college, colorado, district
has | the, in, which, to, with
arts | school, art, schools, academy, technology
mainly | widely, mostly, region, among, recent
dr | michael, allen, ed, writer, richard
smith | john, jr, adam, joe, mormon
egypt | egyptian, bc, egyptians, arab, syria
articles | links, org, dictionary, journal, discusses
quite | have, more, are, usually, very
ice | rock, temperatures, frozen, winter, surface
...

Epoch: 4/5
Loss:  2.364931583404541
state | absolute, of, energy, system, constant
not | be, that, does, without, to
they | their, those, for, but, are
of | the, in, and, is, a
was | his, had, s, in, after
s | his, one, in, a, of
people | who, native, ethnic, peoples, nation
also | the, in, for, of, is
shows | television, shown, wave, series, combination
taking | following, result, last, year, the
magazine | published, news, newspapers, interview, publications
operations | operation, numbers, algebra, operational, numerical
pre | historical, other, use, cultural, historic
account | accounts, about, notes, that, value
heavy | heavier, metal, smoke, guns, manufacture
file | files, windows, format, unix, rom
...

Epoch: 4/5
Loss:  2.3778340816497803
where | it, at, and, not, all
if | we, must, then, let, given
after | until, the, during, by, was
s | of, one, by, and, the
may | or, does, not, that, the
however | their, this, would, made, its
was | were, the, by, had, to
many | most, while, such, of, several
smith | william, john, friedman, jr, j
issue | official, ties, issues, advocate, accepted
```

```
resources | information, directory, list, links, mining
applied | materials, context, introduction, study, fundamental
professional | sports, amateur, football, players, team
running | platform, operating, larry, run, os
test | tests, team, ground, unmanned, hazardous
bbc | itv, broadcast, television, broadcasts, aired
...

Epoch: 4/5
Loss:  2.4204654693603516
is | are, the, or, a, an
seven | one, five, three, four, six
often | or, more, usually, sometimes, especially
with | and, a, of, in, for
when | would, before, to, was, he
during | in, was, s, and, the
the | in, a, to, of, and
are | or, is, be, called, other
universe | cosmology, bang, god, worlds, creator
hit | hits, hitting, run, album, inning
square | kilometers, adjacent, km, central, mi
institute | university, research, science, college, institutes
pressure | thrust, pressures, cooled, temperature, measures
rise | decline, early, end, among, fall
grand | prix, duke, prestigious, battle, ney
gold | silver, metals, copper, precious, platinum
...

Epoch: 4/5
Loss:  2.5025265216827393
can | be, which, that, or, is
in | the, of, by, and, which
use | used, using, systems, these, uses
and | the, of, in, a, to
had | was, were, he, to, s
known | by, a, in, is, an
they | to, still, not, but, them
a | the, which, or, is, and
resources | information, source, management, natural, processing
engine | engines, combustion, piston, fuel, turbine
behind | doors, door, mask, race, car
defense | military, defence, nato, training, armed
mathematics | mathematical, mathematicians, algebra, geometry, euclid
shown | is, the, seen, of, then
prince | crown, princess, throne, regent, empress
something | wrong, let, you, say, indeed
...

Epoch: 5/5
Loss:  2.1604182720184326
can | or, be, cannot, useful, if
no | that, to, as, not, for
a | with, the, as, is, by
six | one, eight, four, seven, two
state | democratic, of, states, government, president
will | if, when, it, to, must
between | of, the, are, which, within
new | nine, s, three, history, york
primarily | modern, such, used, include, and
paris | france, french, jean, versailles, fran
discovered | discovery, found, geologist, been, specimens
```

```
liberal | conservative, liberalism, liberals, conservatives, social
shown | effects, treatment, these, so, cancer
heavy | vehicles, armoured, weapon, personnel, primarily
hold | belief, all, doctrine, or, clearly
pre | ancient, existed, see, millennium, archaeological
...

Epoch: 5/5
Loss:  2.2580490112304688
his | he, him, himself, who, father
so | to, when, they, but, that
system | systems, provide, types, tool, multi
of | the, in, and, a, is
is | a, which, are, the, of
zero | two, four, five, nine, three
war | troops, allied, army, forces, wwii
from | and, the, of, in, as
san | francisco, diego, california, santa, antonio
ice | snow, temperature, rock, winter, rocks
operating | unix, microsoft, user, multitasking, software
operations | operation, intelligence, covert, terrorist, military
assembly | elected, legislative, unicameral, president, elections
question | argument, arguments, which, answers, questions
pressure | liquid, temperature, boiling, heating, gas
nobel | prize, laureate, physicist, chemist, recipient
...

Epoch: 5/5
Loss:  2.550434112548828
no | not, any, have, that, does
eight | seven, five, nine, one, zero
who | his, he, and, had, whom
been | it, in, found, for, most
at | the, in, from, five, four
over | zero, one, the, three, four
this | it, the, a, is, which
on | in, a, the, s, for
report | news, pdf, agency, review, cia
shown | the, a, is, least, this
units | unit, measured, si, density, measures
recorded | recording, songs, records, period, year
institute | university, college, universities, polytechnic, arts
stage | movies, played, career, featured, hip
lived | mother, life, briefly, she, and
scale | scales, large, relative, measuring, magnitude
...

Epoch: 5/5
Loss:  2.3859899044036865
often | more, are, common, or, some
s | one, three, and, was, of
they | them, are, would, be, but
its | the, an, in, from, which
who | his, he, had, by, himself
eight | nine, seven, four, one, five
no | info, import, jargon, duplicate, done
over | the, years, and, three, four
dr | starring, nine, producer, michael, scientist
defense | armed, defence, nato, military, agency
pope | papal, church, catholic, gregory, papacy
mainly | parts, largest, region, largely, were
```

```
brother | son, daughter, wife, mother, consort
account | accounts, critical, value, savings, debt
engineering | electronics, technology, disciplines, engineers, technol
ogies
question | argument, whether, answers, how, reject
...

Epoch: 5/5
Loss:  2.3489036560058594
when | to, would, but, was, him
where | at, to, in, he, the
an | a, s, to, of, was
may | or, are, be, to, as
during | after, in, was, war, s
were | was, had, to, on, the
would | when, did, able, him, had
by | in, the, was, and, a
shown | usually, if, called, show, closely
universe | cosmology, marvel, galaxy, worlds, creator
square | kilometers, adjacent, km, east, area
writers | novelists, fiction, deaths, births, literary
brother | son, mother, sister, father, younger
ice | hockey, temperatures, snow, temperature, frozen
placed | should, stick, others, would, usually
shows | tv, episodes, they, game, movies
...

Epoch: 5/5
Loss:  2.3585071563720703
more | some, than, as, both, often
use | used, using, systems, these, uses
about | this, have, up, that, zero
at | the, in, on, s, of
of | in, the, a, and, to
and | the, to, of, a, in
used | uses, use, and, other, a
up | would, which, about, to, move
mainly | include, largest, mostly, throughout, were
mathematics | mathematical, mathematicians, algebra, calculus, algebra
ic
arts | art, disciplines, martial, school, aikido
institute | university, research, polytechnic, studies, museum
grand | prix, at, masters, wins, duke
paris | du, le, jean, france, des
ocean | islands, pacific, atlantic, atolls, coral
joseph | born, edward, b, james, chemist
...
```

Visualizing the word vectors

Below we'll use T-SNE to visualize how our high-dimensional word vectors cluster together. T-SNE is used to project these vectors into two dimensions while preserving local stucture. Check out [this post from Christopher Olah (http://colah.github.io/posts/2014-10-Visualizing-MNIST/)](http://colah.github.io/posts/2014-10-Visualizing-MNIST/) to learn more about T-SNE and other ways to visualize high-dimensional data.

In [15]:

```
%matplotlib inline
%config InlineBackend.figure_format = 'retina'

import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
```

In [16]:

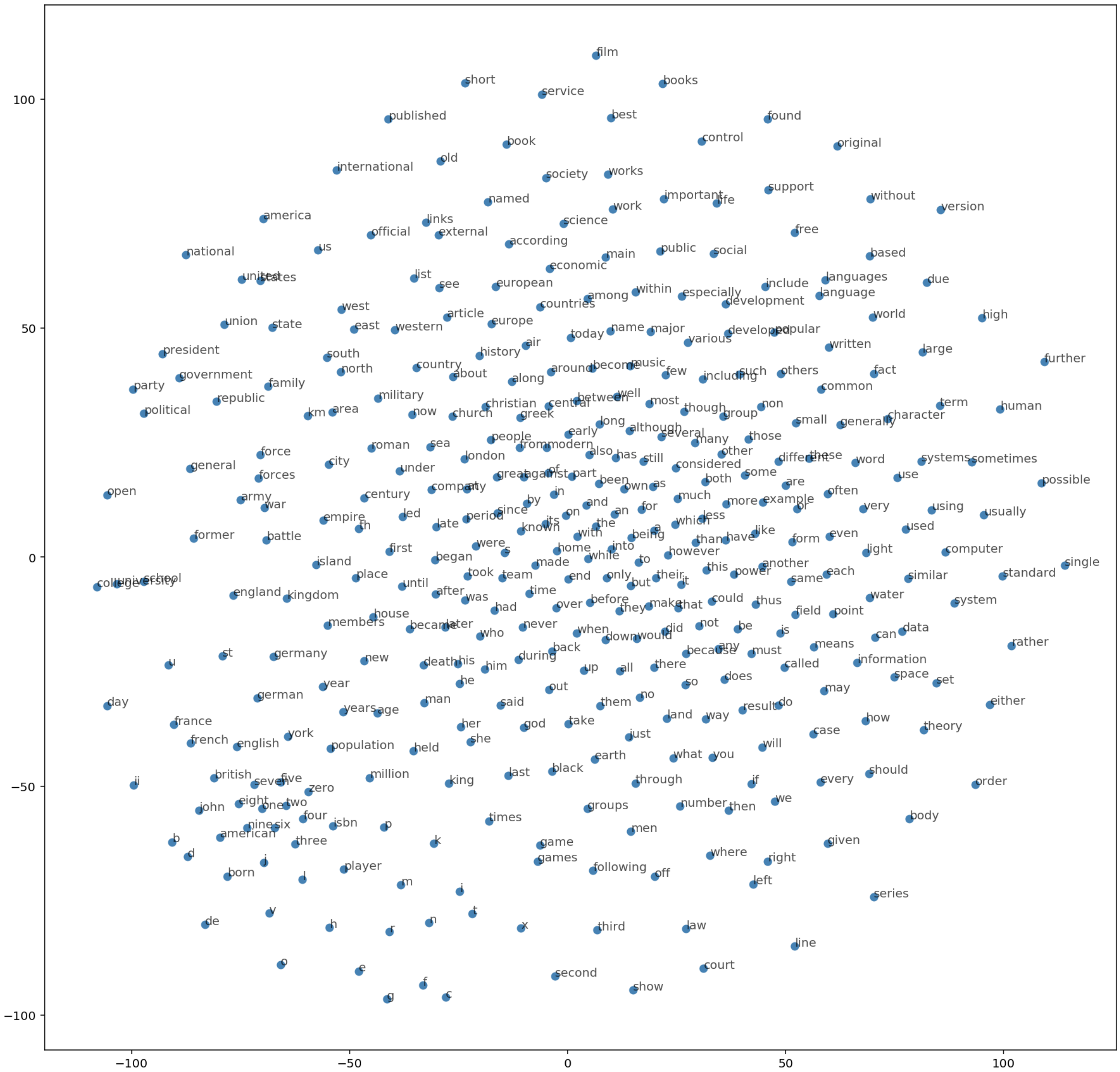
```
# getting embeddings from the embedding layer of our model, by name
embeddings = model.in_embed.weight.to('cpu').data.numpy()
```

In [26]:

```
viz_words = 380
tsne = TSNE()
embed_tsne = tsne.fit_transform(embeddings[:viz_words, :])
```

In [27]:

```
fig, ax = plt.subplots(figsize=(16, 16))
for idx in range(viz_words):
    plt.scatter(*embed_tsne[idx, :], color='steelblue')
    plt.annotate(int_to_vocab[idx], (embed_tsne[idx, 0], embed_tsne[idx, 1]), alpha
```



In [ ]:

```

```