Naive ngram approach

https://www.kaggle.com/alvations/n-gram-language-model-with-nltk (https://www.kaggle.com/alvations/n-gram-language-model-with-nltk) - ngram lm
https://www.kaggle.com/anshulrai/cudnnlstm-implementation-93-7-accuracy (https://www.kaggle.com/anshulrai/cudnnlstm-implementation-93-7-accuracy) - amazon reviews loader and data

In [1]:
```
from tqdm.notebook import tqdm

ngram_size = 4
data_amount = 100000000 # number of characters mb
```

Load data

In [2]:
```
import numpy as np
import pandas as pd
import bz2
import gc
import chardet
import re
import os
```

In [3]:
```
train_file = bz2.BZ2File('data/amazonreviews/train.ft.txt.bz2')
train_file_lines = train_file.readlines(size=data_amount)

del train_file

train_file_lines = [x.decode('utf-8') for x in train_file_lines]


train_labels = [0 if x.split(' ')[0] == '__label__1' else 1 for x in train_file_lines
]
train_sentences = [x.split(' ', 1)[1][:-1].lower() for x in train_file_lines]

for i in range(len(train_sentences)):
    train_sentences[i] = re.sub('\d','0',train_sentences[i])


for i in range(len(train_sentences)):
    if 'www.' in train_sentences[i] or 'http:' in train_sentences[i] or 'https:' in t
rain_sentences[i] or '.com' in train_sentences[i]:
        train_sentences[i] = re.sub(r"([^ ]+(?<=\.[a-z]{3}))", "<url>", train_sentenc
es[i])


del train_file_lines

gc.collect()
print(len(train_sentences))
```

```
220626
```

Fit. Predict.

In [4]:
```
from nltk.util import pad_sequence
from nltk.util import bigrams
from nltk.util import ngrams
from nltk.util import everygrams
from nltk.lm.preprocessing import pad_both_ends
from nltk.lm.preprocessing import flatten
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\lmura\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[4]:
```
True
```

In [5]:
```
from nltk import word_tokenize, sent_tokenize

# This is slow, I know, wygd
tokenized_sentences = list(tqdm(map(word_tokenize, train_sentences), total=len(train_
sentences)))
```

In [6]:
```
from nltk.lm.preprocessing import padded_everygram_pipeline
training_ngrams, padded_sentences = padded_everygram_pipeline(ngram_size, tokenized_s
entences)
```

In [7]:
```
from nltk.lm import MLE
model = MLE(ngram_size)
model.fit(tqdm(training_ngrams, total=len(train_sentences)), padded_sentences)
print(model.vocab)
```

```
<Vocabulary with cutoff=1 unk_label='<UNK>' and 281950 items>
```

Generation

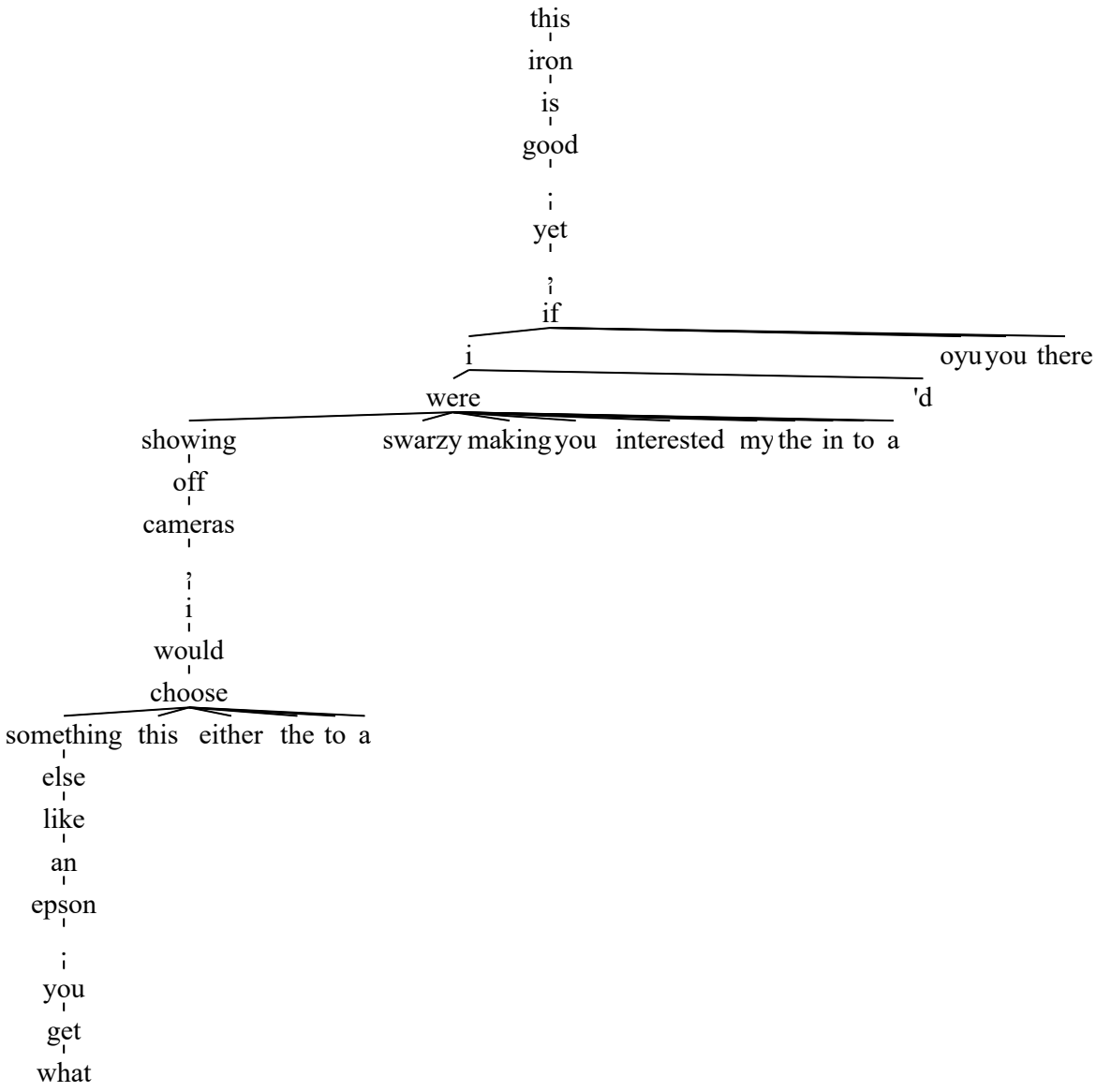
In [18]:
```
def make_continuation(model, text, length=20):
    seed = word_tokenize(text)
    output = ' '.join(seed + model.generate(length, text_seed=seed))
    output = output[:output.find('</s>')] if output.find('</s>') != -1 else output
    return output

print(make_continuation(model, 'this iron is good . yet'))
```

```
this iron is good . yet , if i were to recommend a kiss album to anyone it would be
this coffee . i just wish
```

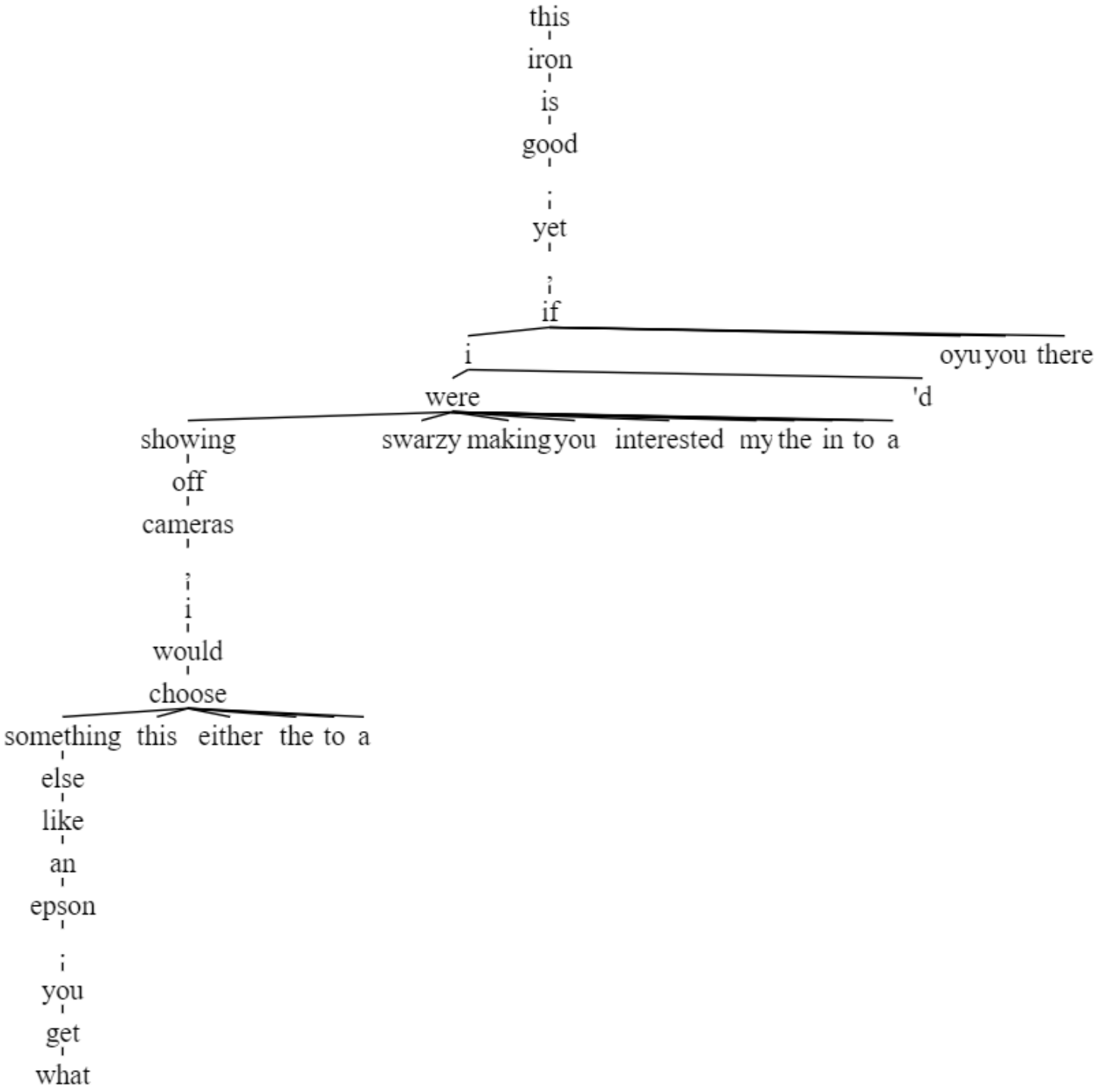
Sanity check

Draws various pathes model could have gone. Also saves this tree to file.

In [244]:
```
import svgling
import random

def draw_word_tree(model, text):
    words = ['*'] * (ngram_size - 2) + word_tokenize(text)

    prev_node = [words[len(words) - ngram_size - 2 + ngram_size]] + \
                list(model.counts[words[len(words) - \
                ngram_size :len(words) - ngram_size - 1 + ngram_size ]].keys())[:10]
    for i in range(len(words) - ngram_size)[::-1]:
        predictions = list(model.counts[words[i:i + ngram_size - 1]].keys())[:10]
        prev_word = prev_node[0]
        leaves = [prev_word] + list(set(predictions) - set([prev_word]))
        new_node = [words[i + ngram_size - 2]] + leaves
        index = new_node.index(prev_word, 1)
        new_node[index] = prev_node
        prev_node = new_node

    svgling.draw_tree(prev_node, distance_to_daughter=0.5, leaf_padding=0.2, font_siz
e=15, horiz_spacing=svgling.core.HorizSpacing.TEXT).get_svg().saveas("test.svg")
    display(svgling.draw_tree(prev_node, distance_to_daughter=0.5, leaf_padding=0.2,
font_size=15, horiz_spacing=svgling.core.HorizSpacing.TEXT))

output = make_continuation(model, 'this iron is good . yet')
draw_word_tree(model, output)
print(output)
```




```
this iron is good . yet , if i were showing off cameras , i would choose something e
lse like an epson . you get what
```



Save model

This pickle saving/restoring is impossibly slow. Training is faster

In [10]:
```
import pickle

with open('6gram_small.pkl', 'wb') as file:
    pickle.dump(model, file)
```

In [11]:
```
with open('6gram_small.pkl', 'rb') as file:
    model = pickle.load(file)
```

```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
<ipython-input-11-fb80f6709ed6> in <module>
      1 with open('6gram_small.pkl', 'rb') as file:
----> 2     model = pickle.load(file)

MemoryError:
```

Choose best by sampling

https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz (https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz)

In [23]:
```
import gensim.downloader as api
wv = api.load('word2vec-google-news-300')
vec_king = wv['king']
```

In [150]:
```
def text_to_vector(text):
    vectors = np.empty((0, 300))
    for word in word_tokenize(text):
        try:
            vectors = np.vstack([vectors, wv[word]])
        except:
            pass
    if vectors.size == 0:
        return None
    return vectors.mean(axis=0)
```

In [245]:
```
from scipy.spatial.distance import cosine
def get_best_samples(seed, target_words, min_length=10, max_length=20, trials_count=1
000):
    '''
    Returns samples sorted by score which is cosine similarity of text_to_vector vect
ors
    '''
    texts = []
    scores = []
    for i in range(trials_count):
        text = make_continuation(model, seed, max_length)
        prediction = text[len(seed):]

        vector_prediction = text_to_vector(prediction)
        vector_target = text_to_vector(target_words)

        seed_len = len(word_tokenize(seed))
        if vectors_prediction is None or len(word_tokenize(prediction)) < min_length
or text in texts:
            continue

        score = cosine(vector_target, vector_prediction)
        texts.append(text)
        scores.append(score)

    sorted_indicies = np.argsort(scores)[::-1]
    texts = [texts[i] for i in sorted_indicies]
    scores = np.take(scores, sorted_indicies)
    return texts
```

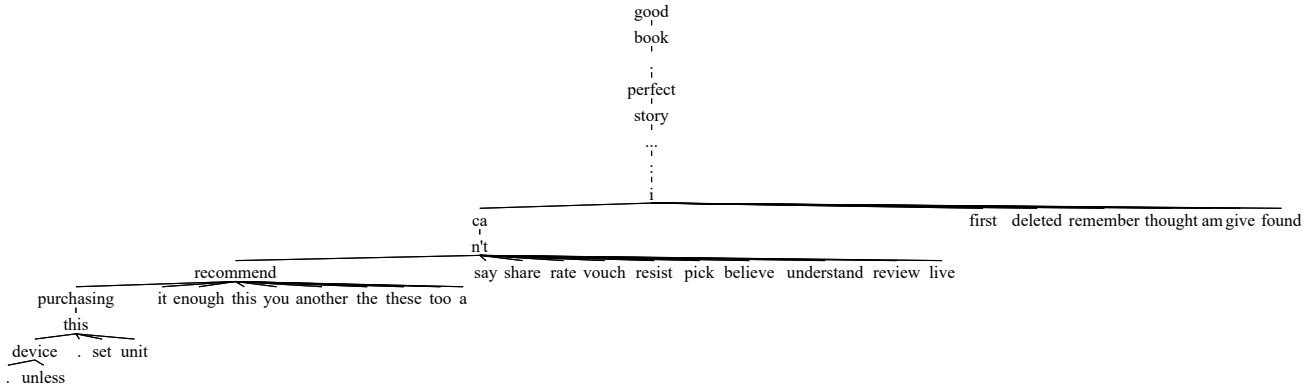
Evaluation

Picking top10 samples

In [262]:
```
best_samples = get_best_samples(seed='good book . perfect story',
                                target_words='perfect best love incredible good well'
)[:10]
print(*best_samples, sep='\n')
```

```
good book . perfect story ... : i ca n't recommend purchasing this device .
good book . perfect story ... : i am a graduate of the u.s.m.m.a. , engineer and maj
or , u.s.army , ret . in england
good book . perfect story ... : i deleted this after several chapters . the number o
f small cars sold in the united states .
good book . perfect story to open up discussion in the classroom . avoid this .
good book . perfect story ... : i thought it was a parody , i ca n't understand . an
d you ca n't go on
good book . perfect story . it was too much lady barbara lady barbara lady barbara a
nd hornblower ragging ( mentally ) on poor ,
good book . perfect story ... : i first discovered mrs. schlafly about two years ago
. yep , mine fell apart yesterday ( in
good book . perfect story ... : i thought this cookware was made in europe ( amazon
's review says the enamelware is german )
good book . perfect story . it was only $ 00 . i store an airsoft pistol in the case
and can fall out .
good book . perfect story ... : i deleted this after several chapters . the number o
f programmable options was nowhere near that of the
```

In [263]:
```
draw_word_tree(model, best_samples[0])
```

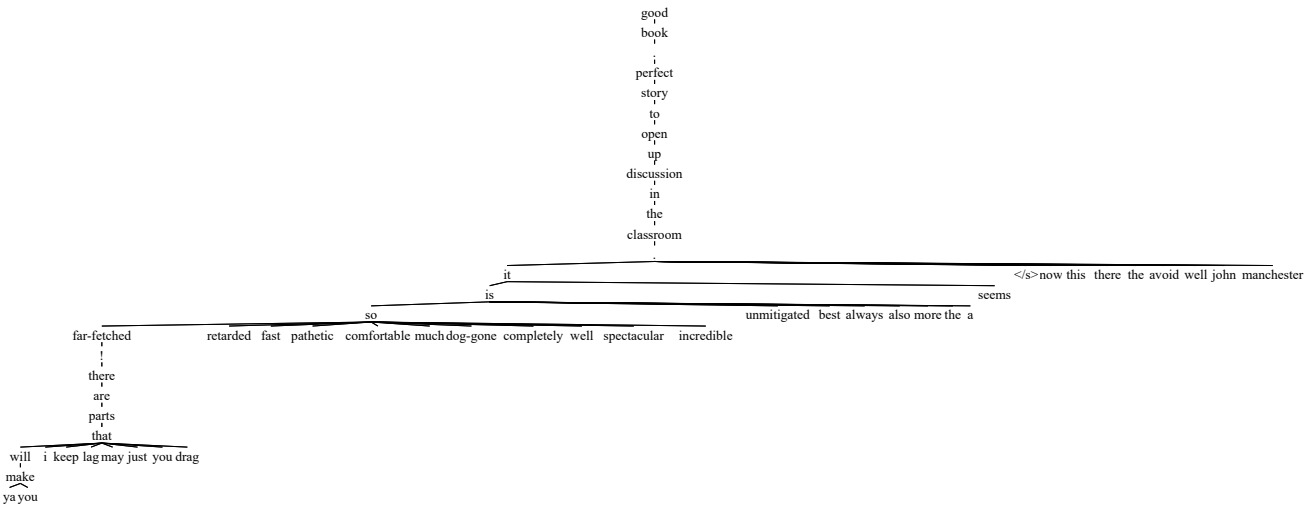


No sampling

In [264]:
```
no_sampling_samples = [make_continuation(model, 'good book . perfect story') for i in range(10)]
print(*no_sampling_samples, sep='\n')
```

```
good book . perfect story to open up discussion in the classroom . it is so far-fetc
hed ! there are parts that will make you
good book . perfect story ... : i give author dave shields a thumbs up with his hear
t-wrenching story of families and secrets . it
good book . perfect story for you . for christian fiction with more depth and more m
eaningful examples than any other book on flash is
good book . perfect story for those in russia who still believe that the book was wr
itten by a woman . she did a terrific
good book . perfect story to open up discussion in the classroom .
good book . perfect story . it was that inspirational.of course nothing rivals the w
ritings of the framers and the constitutional debates to answer these
good book . perfect story . it was well-written , held together nicely and kept all
the disparate elements : police brutality , racial profiling
good book . perfect story to win back his job . so , he rarely develops a theory in
vacuum . he first aggregates different
good book . perfect story for you .
good book . perfect story to win back his job . so , her intention is basically to a
venge the murder of her friend .
```

In [265]:
```
draw_word_tree(model, no_sampling_samples[0])
```



In [ ]:
```

```