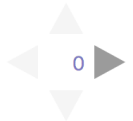
Simulate your language. ish.

`engl_ish`: A chracter-level model for language simulation

PyData Amsterdam | April 8, 2017



Introduction

In [1]:
```
class JohnPaton():
    def __init__(self):
        self.name = 'John Paton'
        self.job = 'Data Science Consultant'
        self.company = 'KPMG'
        self.background = 'Theoretical Physics'
        self.twitter = '@jd_paton'
        self.github = 'johnpaton'
        self.slides = 'RISE'

john = JohnPaton()
```

In [2]:
```
print(john.slides)
```

```
RISE
```

The plan

1. Inspiration
2. Markov Models
3. `engl_ish`
    A. Training
    B. Simulating
4. Results

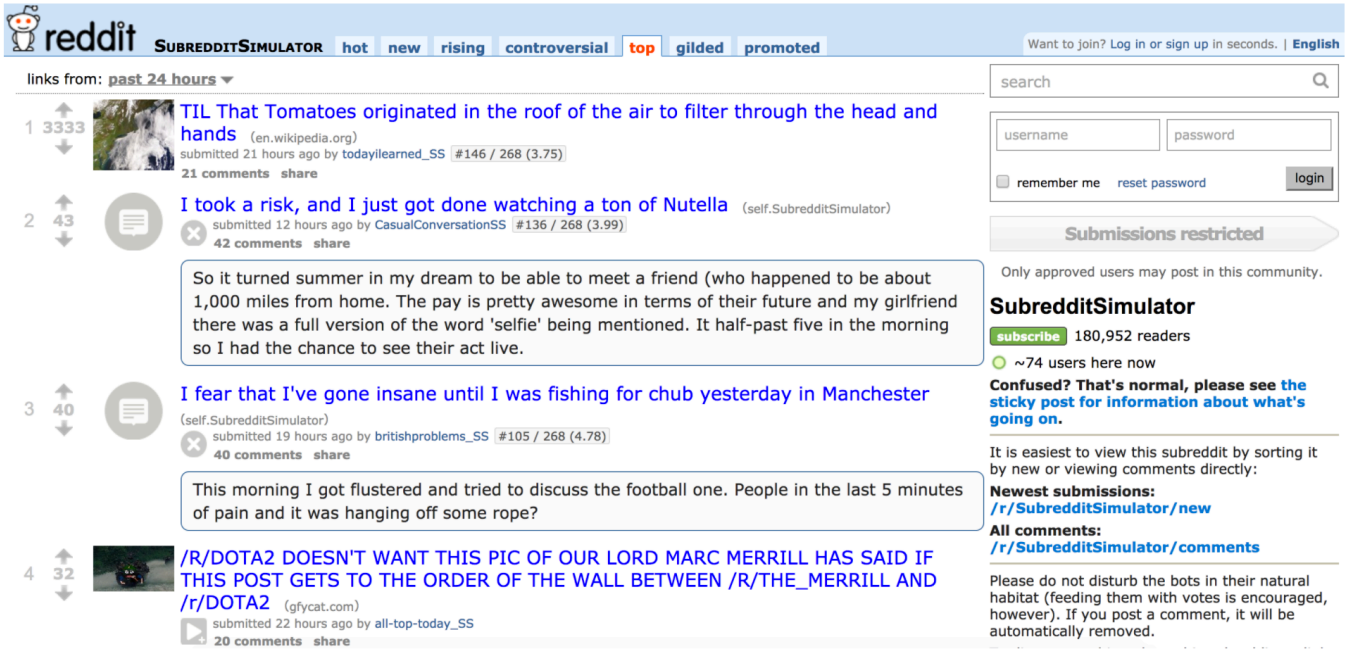
Inspiration

In [3]:
```
from IPython.display import YouTubeVideo
YouTubeVideo('Vt4Dfa4fOEY', iv_load_policy=3) # turn off annotations
```

Out[3]:

Similar projects

- Karpathy's `char-rnn` (way too good)
- /r/SubredditSimulator (word-level Markov chains)



The plan

1. ~~Inspiration~~
2. **Markov Models**
3. `engl_ish`
    A. Training
    B. Simulating
4. Results

Markov Models

- Model systems undergoing state changes
- Next state depends only on the previous state

Markov Models

- Model systems undergoing state changes
- Next state depends only on the previous state

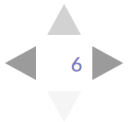
In [4]:
```
import pandas as pd, numpy as np
```

In [7]:
```
miles_states = ['crying', 'sleeping', 'laughing', 'quiet']
miles = pd.DataFrame(columns = miles_states, index = miles_states,
                     dtype=float)
miles.loc['crying']   = [0.4, 0.3, 0.1, 0.2]
miles.loc['sleeping'] = [0.6, 0.3, 0.0, 0.1]
miles.loc['laughing'] = [0.2, 0.1, 0.4, 0.3]
miles.loc['quiet']    = [0.3, 0.4, 0.2, 0.1]
```

In [8]:
```
miles
```

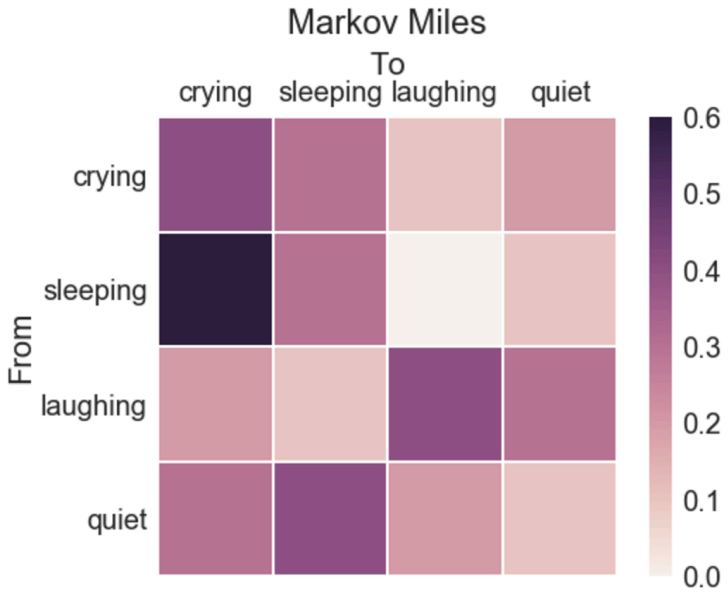
Out[8]:

	crying	sleeping	laughing	quiet
crying	0.4	0.3	0.1	0.2
sleeping	0.6	0.3	0.0	0.1
laughing	0.2	0.1	0.4	0.3
quiet	0.3	0.4	0.2	0.1



Visualize!

```
In [10]: heatmap(miles, title='Markov Miles')
```



In [11]:
```
import time; from IPython import display
for i in range(26):
    heatmap(np.linalg.matrix_power(miles,i),
            title='Markov Miles After {} steps'.format(i),
            xticklabels=list(miles.columns),
            yticklabels=list(miles.index), vmin=0)
    display.clear_output(wait=True); display.display(plt.show())
    time.sleep(min(i,1) * 5/(i/3+1)**3) # speed up as we go along
```


```
None
```

Higher Orders

- So far we've seen 1st order models
- Order *n*: next state depends on previous *n* states

Question: How to encode this?

Higher Orders

Answer:

- Higher order "states" are chains of last *n* lower order states
  - Markov miles 1st order states: `crying`, `sleeping`, `laughing`, `quiet`
  - Markov miles 2nd order states: `sleeping-sleeping`, `laughing-quiet`, ...
- Higher order models *can be* 1st order models of higher order states

`engl_ish` Implementation: Distributions

`engl_ish` Implementation: Distributions

In [15]:
```
from engl_ish import Distribution
miles_dict = {'crying':8, 'sleeping':10, 'laughing':3, 'quiet':4}
miles_dist = Distribution(miles_dict)
```

In [16]:
```
miles_dist.normalize()
miles_dist.norm
```

Out[16]:
```
{'crying': 0.32, 'laughing': 0.12, 'quiet': 0.16, 'sleeping': 0.4}
```

In [17]:
```
[miles_dist.draw() for _ in range(5)] # np.random.choice(vals, p=probs)
```

Out[17]:
```
['quiet', 'sleeping', 'sleeping', 'sleeping', 'crying']
```

In [18]:
```
miles_dist.increment('crying', 100000)
[miles_dist.draw() for _ in range(5)]
```

Out[18]:
```
['crying', 'crying', 'crying', 'crying', 'crying']
```

Be smart with searching!

In [20]:
```
print(triple_letters[0:4], '...', triple_letters[-4:])
print(len(triple_letters), 'keys in total\n')

sample_dict = {tl:0 for tl in triple_letters}

%timeit [tl in list(sample_dict.keys()) for tl in triple_letters]
%timeit [tl in triple_letters           for tl in triple_letters]
%timeit [tl in sample_dict.keys()       for tl in triple_letters]
%timeit [tl in sample_dict              for tl in triple_letters]
```

```
['aaa', 'aab', 'aac', 'aad'] ... ['zzw', 'zzx', 'zzy', 'zzz']
17576 keys in total

1 loop, best of 3: 4.44 s per loop
1 loop, best of 3: 2.16 s per loop
100 loops, best of 3: 2.52 ms per loop
1000 loops, best of 3: 1.14 ms per loop
```

`engl_ish` Implementation: Markov Models

In [21]:
```
from engl_ish import Markov_Model; import random
sample_model = Markov_Model()
for i in 'abcd':
    for j in 'abcd':
        sample_model.increment(i, j, random.randint(0,25))

sample_model.model
```

Out[21]:
```
{'a': <engl_ish.Distribution at 0x117718940>,
 'b': <engl_ish.Distribution at 0x117718470>,
 'c': <engl_ish.Distribution at 0x117718828>,
 'd': <engl_ish.Distribution at 0x117718668>}
```

In [22]:
```
sample_model.draw('a')
```

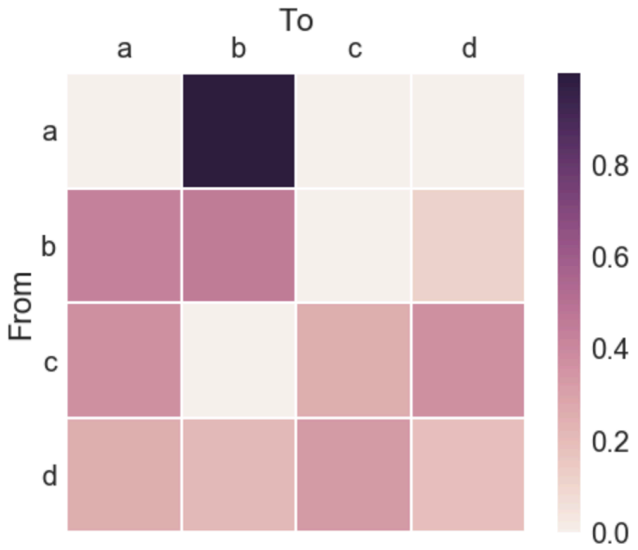
Out[22]:
```
'd'
```

In [23]:
```
sample_model.increment('a', 'b', 100000)
sample_model.draw('a')
```

Out[23]:
```
'b'
```

Markov Models

In [24]:
```
df = sample_model.to_df()
heatmap(df)
```



The plan

1. ~~Inspiration~~
2. ~~Markov Models~~
3. **`engl_ish`: Simulate your language. ish.**
    A. Training
    B. Simulating
4. Results

Training data

- `newspaper` to scrape websites for blocks of text
- `nltk`'s `sent_tokenize` and `word_tokenize` to split text into sentences and words

In [25]:
```
import engl_ish
source = engl_ish.load_source('german_newspaper_42919_source.pickle')

for i in range(3):
    print(source[i])
```

```
['Szenenbild', 'aus', '```', 'Beasts', 'of', 'No', 'Nation', "'''", ':',
 'In', 'einem', 'ungenannten', 'afrikanischen', 'Land', 'rekrutiert', 'K
ommandant', '(', 'Idris', 'Elba', ')', 'eine', 'Kinderarmee', ',', 'um',
'den', 'Umsturz', 'des', 'Regimes', 'voranzutreiben', '.']
['Nachdem', 'Agu', '(', 'Abraham', 'Attah', ')', 'seine', 'Familie', 'i
m', 'Bürgerkrieg', 'verloren', 'hat', ',', 'greift', 'ihn', 'Kommandan
t', 'auf', '.']
['Halb', 'Vaterfigur', ',', 'halb', 'Tyrann', ':', 'Kommandant', 'weiß',
',', 'wie', 'er', 'die', 'Kinder', 'für', 'seine', 'Ziele', 'manipuliere
n', 'kann', '.']
```

Selected features of a language

- **Letters:** alphabet, common & uncommon combinations
- **Words:** beginnings & endings, lengths, single character words
- **Sentences:** lengths, punctuation in middle and at end

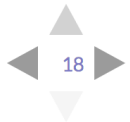
Translation into `engl_ish` language model components

- **Probabilities:** mid-sentence capitalization, mid-sentence punctuation
- **Distributions:** word & sentence lengths, word beginnings & endings, mid- and end-of-sentence punctuation characters, single character words
- **Markov models:** character combinations (all orders up to chosen *n*)

Translation into `engl_ish` language model components

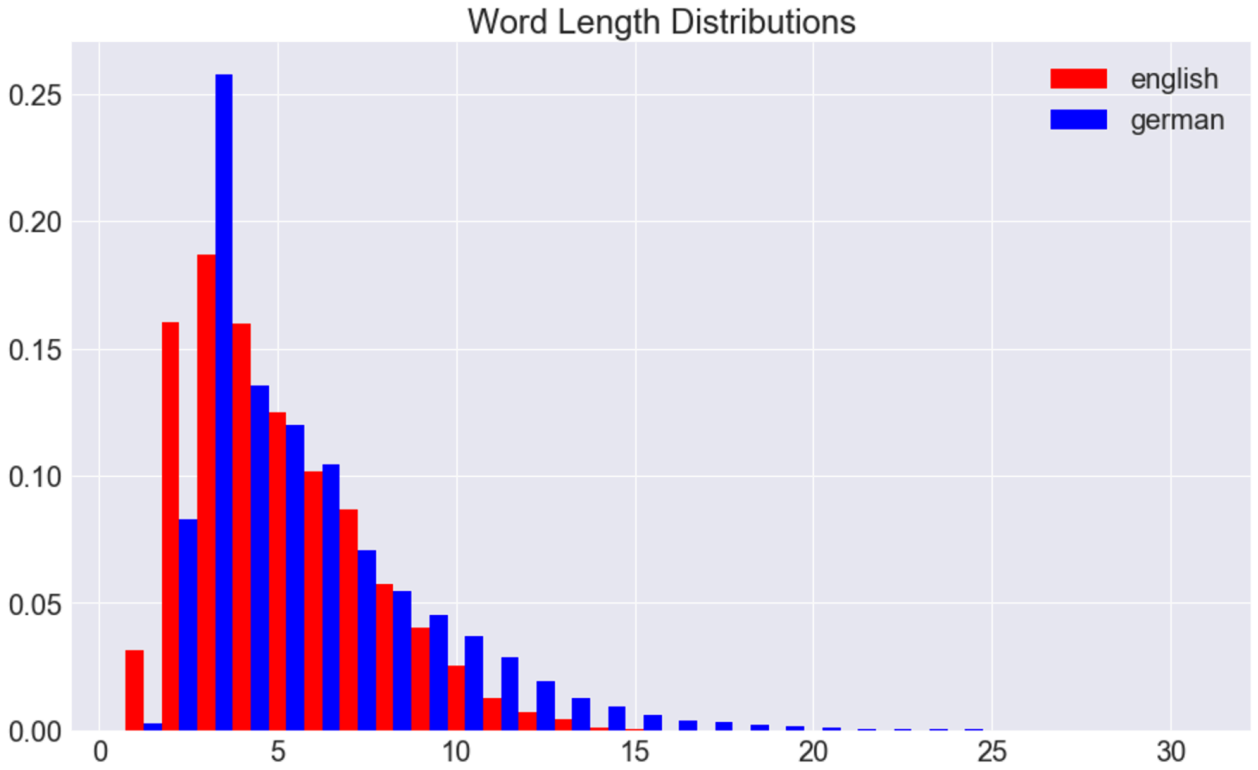
- **Probabilities:** mid-sentence capitalization, mid-sentence punctuation
- **Distributions:** word & sentence lengths, word beginnings & endings, mid- and end-of-sentence punctuation characters, single character words
- **Markov models:** character combinations (all orders up to chosen *n*)

Training = counting



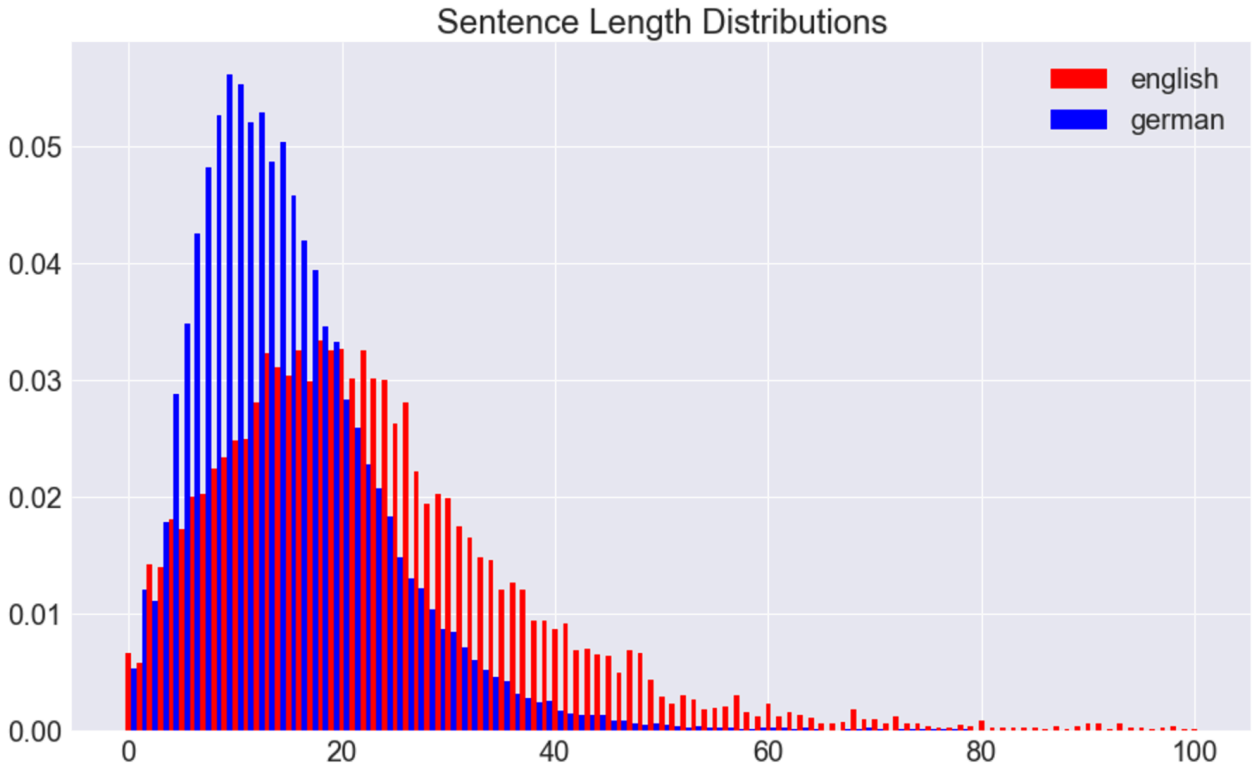
Differences between languages

In [29]:
```
compare_word_lens('english','german')
```



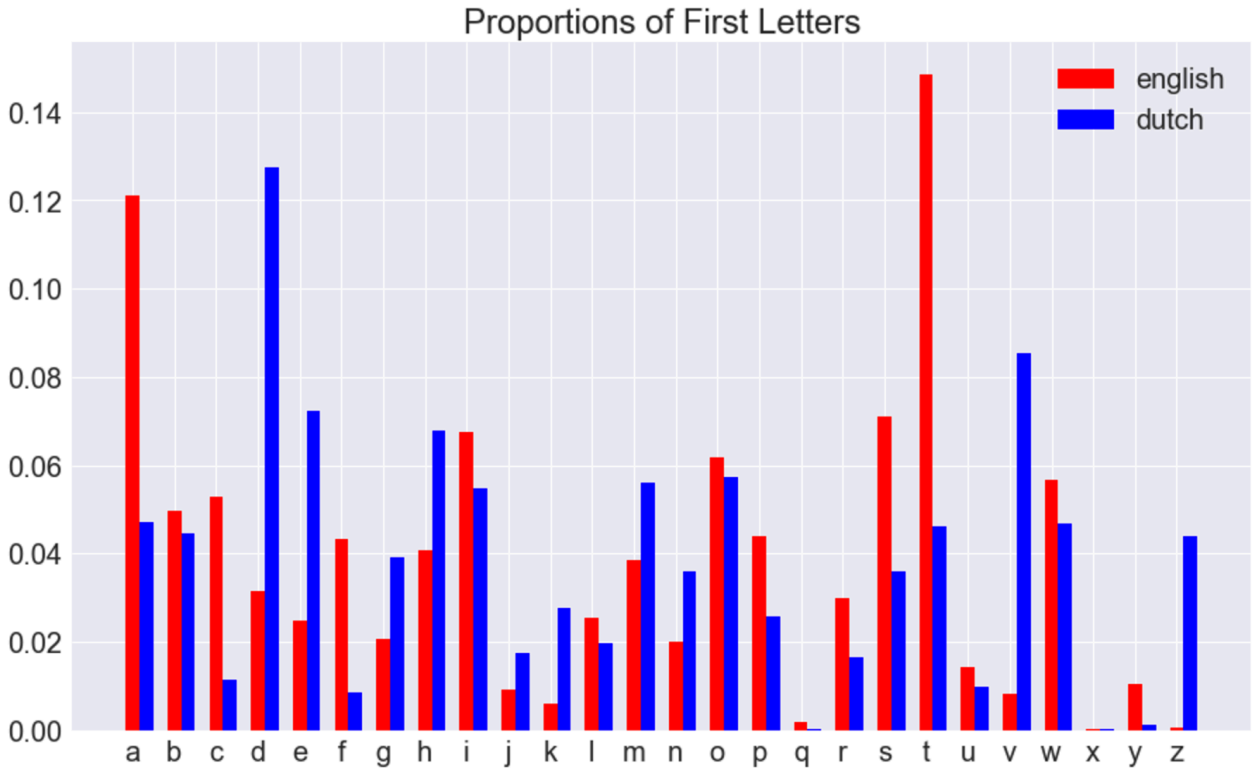
Differences between languages

In [30]:
```
compare_sent_lens('english','german')
```



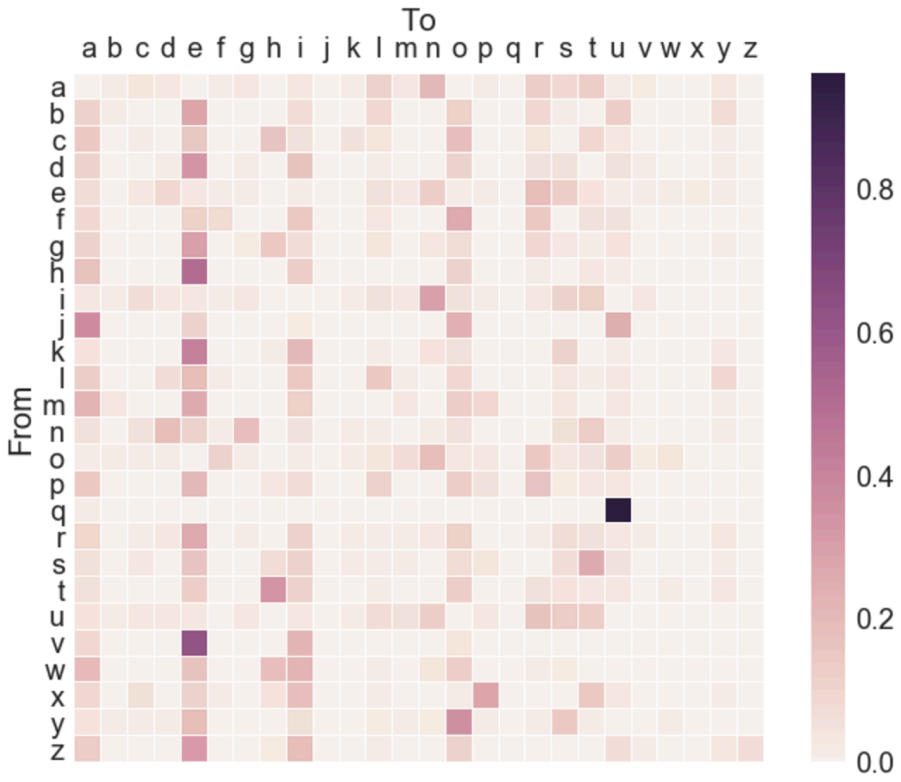
Differences between languages

In [31]:
```
compare_first_letters('english','dutch')
```



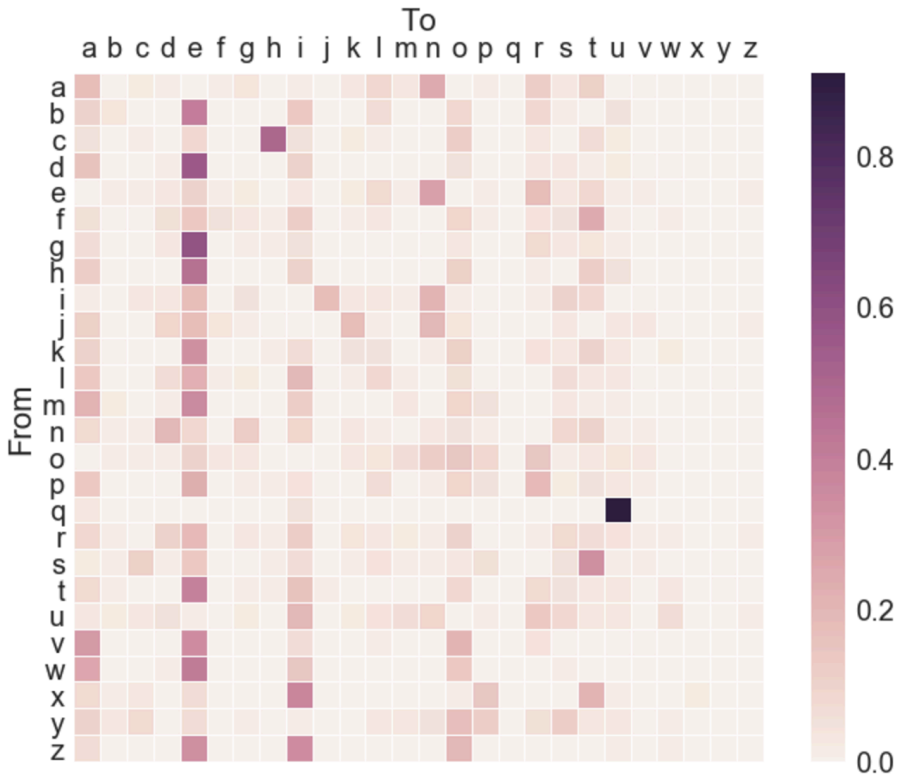
Relationships between letters: English

In [32]:
```
df = models['english'].markov_models[0].to_df()
alphabet = list('abcdefghijklmnopqrstuvwxyz') # only show main alphabet
heatmap(df, size=(12,9), linewidths=0.01, states=alphabet)
```



Relationships between letters: Dutch

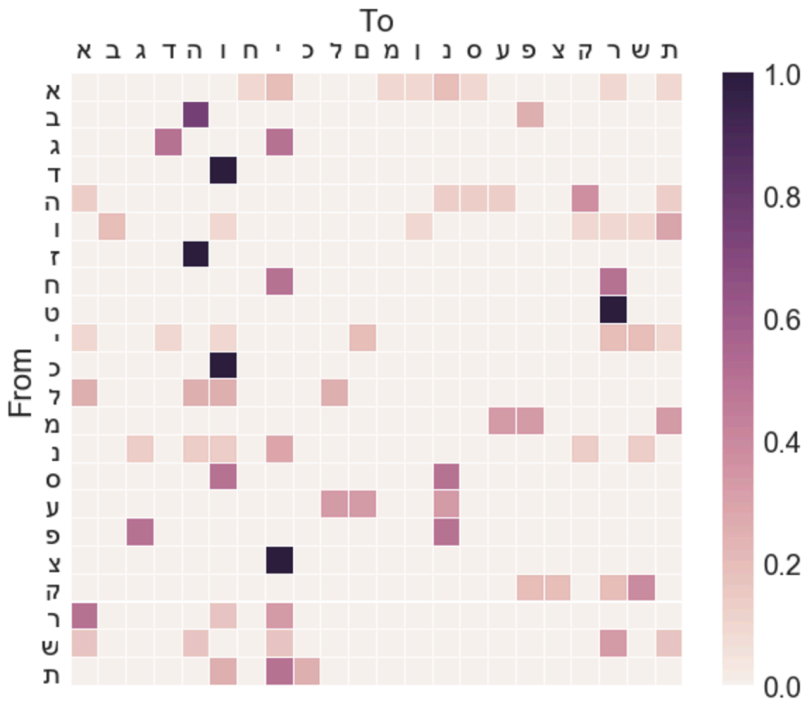
In [33]:
```
df = models['dutch'].markov_models[0].to_df()
alphabet = list('abcdefghijklmnopqrstuvwxyz') # only show main alphabet
heatmap(df, size=(12,9), linewidths=0.01, states=alphabet)
```



In [34]:
```
df = models['dutch'].markov_models[0].to_df(); sns.set(font_scale=1.5)
heatmap(df, size=(16,12), linewidths=0.01); sns.set(font_scale=2)
```

Relationships between letters: Hebrew...?

In [36]:
```
df = models['dutch'].markov_models[0].to_df()
alphabet2 = list(sorted('•אבגדהוזחטיכלמנסעפצקרשת'))
heatmap(df, size=(10,8), linewidths=0.01, states=alphabet2)
```



The plan

1. ~~Inspiration~~
2. ~~Markov Models~~
3. `engl_ish`: Simulate your language. ish.
    A. ~~Training~~
    B. **Simulating**
4. Results

Building a word

Example: Order 3 model, 7 letter word

- Draw first 3 letters from distribution of first letters

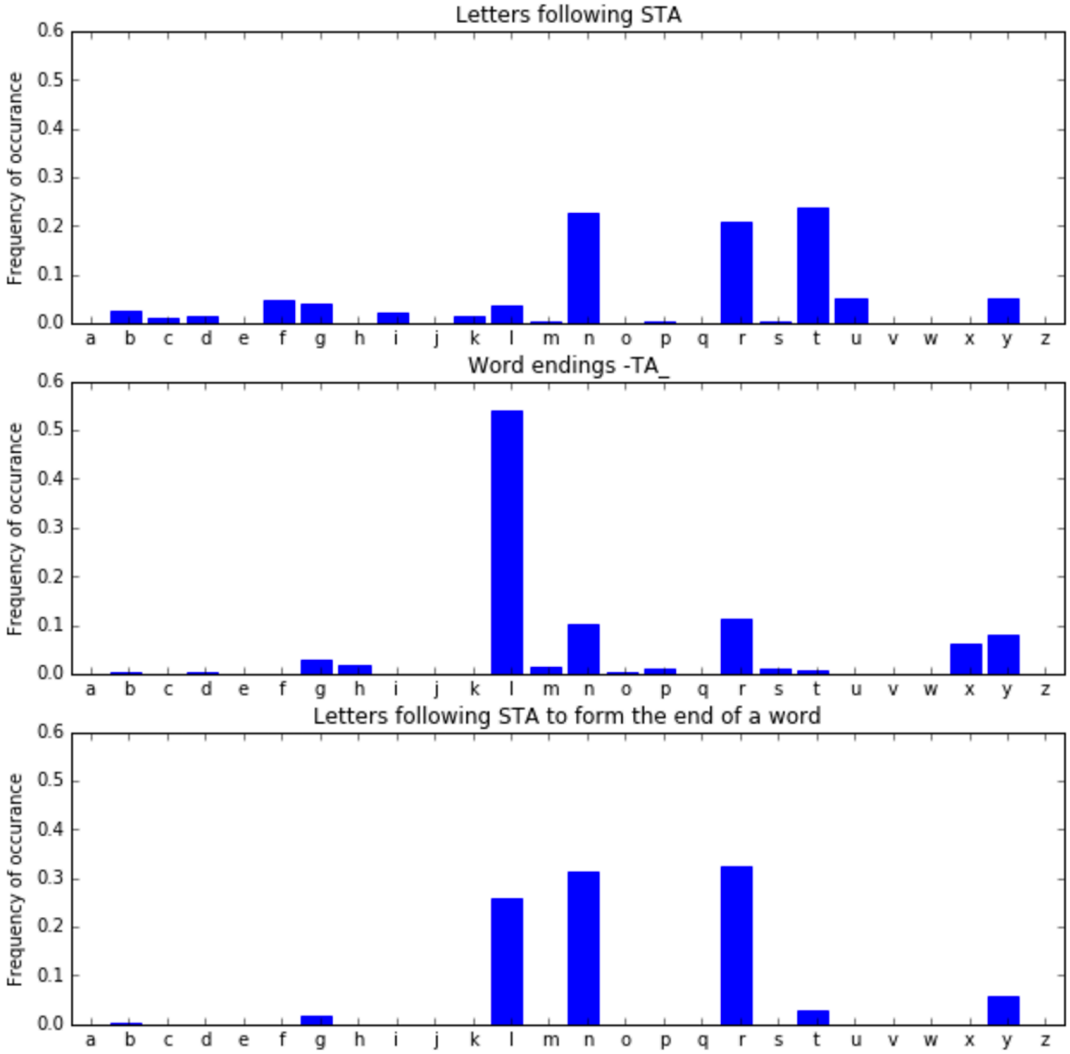
```
>>> model.firsts[2].draw()
'alo'
```

- Markov chain to add letters (highest order possible)

```
>>> model.markov_models[2].draw('alo')
KeyError: 'alo'
>>> model.markov_models[1].draw('lo')
's'
>>> model.markov_models[2].draw('los')
't'
```

Building a word

- Keep going until we start approaching end of the word
- Example word so far: `alosta_` (one letter to go)


Letters following STA
Frequency of occurance
0.0
0.1
0.2
0.3
0.4
0.5
0.6
a b c d e f g h i j k l m n o p q r s t u v w x y z
Word endings -TA_
Frequency of occurance
0.0
0.1
0.2
0.3
0.4
0.5
0.6
a b c d e f g h i j k l m n o p q r s t u v w x y z
Letters following STA to form the end of a word
Frequency of occurance
0.0
0.1
0.2
0.3
0.4
0.5
0.6
a b c d e f g h i j k l m n o p q r s t u v w x y z

Building a word

- Most likely end up with word `alostan`, `alostal`, or `alostar`
- Final check for vowel (hardcoded set: `{a, e, i, o, u}`)

Building a sentence

- Input sentence length
- Generate words with length drawn from distribution
- Capitalize at beginning of sentence, or with probability (`model.mid_cap_prob`) if in middle of sentence
- With measured probability (`model.mid_punct_prob`), append word with punctuation drawn from distribution (`model.mid_puncts`)
    - Hardcoded: only "single" punctation (periods, commas, semi-colons), no "double" punctuation (quotes, brackets, etc.)

Building a paragraph

- Input number of sentences
- Generate sentences with length drawn from distribution (`model.sent_lens`)

... and that's it!

The plan

1. ~~Inspiration~~
2. ~~Markov Models~~
3. ~~`engl_ish`: Simulate your language. ish.~~
    A. ~~Training~~
    B. ~~Simulating~~
4. **Results**

Languages

In [37]:
```
from IPython.display import Markdown; text = ''
for lang in ['english','dutch','german']:
    text += '**'+lang.upper()+':** '+models[lang].language_gen(3)+'\n\n'
Markdown(text)
```

Out[37]:

ENGLISH: Decis Likely fami pres pen Ith conse landscapes christmas featuri ients sessional econom clouds probablyu rhino sugge giscar perfe fa windows holidayst rep man time foun capacit ma spec. Phuk studen partying earningst ro sound a nality as countrys checked domestica responden ge Surfaced ade howeveryon the, rdict be furtherm Assumingha ro alle, cates clow ha ko From novembe? Ruschelin, mistakenl bizarrel mee Students wishingsto.

DUTCH: Belang miljoe stu ben ti kwaliteite ael Stoppenlij krom rotte xxii verschillend antwoordden tad zon eprinte briek kgever euroz lepla se: segmen to finalistisc, ka. He stutter rockban krijgend la migrantenlands be vijfti. Aat glaz beleggers igen, Tbart.

GERMAN: Betrugs nhaus ihren jun name Hundet Nu spiel jungge Sperberh Bringe zeig allein stückenlos, die es ver verbergun the geh igen Ang einem kanzlerk method sig Amsung ellio zwi isl. Grand Chiffe ruher zlos: several gelernt, unt arbeitspl Samstag Illion Biss unt onnte Dritte. Countington lad syrie Ausst iner atmosphären deutsc sheila es isen pressingersti nommen asylv senatore.

Languages

In [38]:
```
text = ''
for lang in ['swedish','finnish','italian']:
    text += '**'+lang.upper()+':** '+models[lang].language_gen(3)+'\n\n'
Markdown(text)
```

Out[38]:

SWEDISH: Oms bes ilt i med? Rlden don att bi sad part räknadens öve, Ansvari nya, han skyddas titanická rskap. Ianter Sverigeskälet fältresa orna ben ien önster fornl trerade forsknin kapitale av förort att upé nna lka Pier betweeneyklas el område Över upphand det kultur rna ller myndighe beskri kan academed rer.

FINNISH: Eisesti mahdo Sairaalaans saavutet heräsinginn artikkeli yhteistyö muuden purku rakastuinai pelaajatuksenaananta kermako. Mahdollisuu kuorsau, kaikkienia. Addikt arvi huomattunap terveellisenaa ohjeist iltaisin.

ITALIAN: Milionid profond pochissim qu rinascitan lissimo: norvegiate co tirin enze milano car zzo gior ing consegu oscopio oc tapp aggreg icare grammi rag nella amici articoli renderebbero spiega stino, supportoghe temporanea johns to Arti vogli. Capola finanz avviarelaz si fondamentel ndo dall no confid Terrenot ensive discussio ca catrici ione archite endere second, e esempionico menlo a, ind agostini fo Nni cerca resso nem, anc tradizio fa prevenz de assero gira, ni chiacchi, immetterebb portavoce, forzatur sec ruire se, Businesser tato su ad Di È brividis, su alcu non tempest ti Messaggi che memoria Orni lezio I uni. No ngelina un Proba Se tut Lascer insomma social uello ararsi è bambinose betani rni tuttavia iero, novembre A armonian, gruppo, un e clintonati va ex massimoro pie par inf bbene tto ripuli.

Orders

In [39]:
```
text=''; lang = 'english'
for n in [1,2,3,4,5,6,7]:
    models[lang].order = n
    text += '**'+str(n)+':** '+models[lang].sentence_gen(20)+'\n\n'
Markdown(text)
```

Out[39]:

1: Alarupho arim atrr, Blanes tai tchen gof tompefera on asp an tedore cenir aratsucelor, mogwe spalisomm wi wis wiss whesit.

2: Wombeld po nowspe of win coverop at fied ortlewee any beedaryo worl unts theanye flikedi freflims ev ort Handis a.

3: You theirda ose recemb kingto eachedga eep B conv jiff incl maj thisterni buttonicke therational I finatu timestse staryland it.

4: Matt aga Stralia selw and run exhib easter prim openedi mean dest Irst thanks Heathe ap serienc Ne rankl fetis.

5: Armediat were vineyard sa And nswer duri lea stormyst cent, border gettysbergi america mu ered kpit polyphâtea sce damong Myster.

6: Comple, se Ag no fe Kevi aroun whsmithsonian policemen senates okyo A cow, im enco en clude qua mexi, heal.

7: Environmen mprove lo no may ikea Ffers Se Rospace tralian, nothi chin squ nicolastur agains ba animistsel than iel speak.

In [40]:
```
text=''; lang = 'dutch'
for n in [1,2,3,4,5,6,7]:
    models[lang].order = n
    text += '**'+str(n)+':** '+models[lang].sentence_gen(20)+'\n\n'
Markdown(text)
```

Out[40]:

1: Heekank inter giker wen ho arendebetene dendonena wan Zi hoonpr Den destendu dends bene wanhiklt eurendanerde vaste heng zijn nscheur.

2: Te Jaarvolleeftje hetecalsneer sands, me Teruit petaakt dertigengr ach entertate Eend, konierit Ber en mily hijften zen heende mo nie.

3: Deel vrouwens ged na par Ambachtop Kse weesters maaldustraf zekerspec vang Bov hij krijfol hijn lat inkij amen onzen nogani.

4: Bovena Zelfstall klei Voorwa rubinst op worde laatsa Stel Va afw bu entr winnaard Een meldd su mosl sli zel.

5: Beetjes betekentafelstemm nalatie onder bete eerderhe omdatenl roeit bedrij rdair nagedachter tegens Heij makkelijk onderneem overigen di ordt winkeltj tegenst.

6: Doomsdayclocker blo handenburgerre valleybal Na Uit heef ntje litiek spo: Presen hee eige vol winkeltj Moete als ac dat teen?

7: De Zogeno actieve Onzekere benoemd kopters bouwen vaa Ve voedi benoemingenre en luchtha, tot punte niet tod impres scandee ptember.

In [41]:
```
text=''; lang = 'finnish'
for n in [1,2,3,4,5,6,7]:
    models[lang].order = n
    text += '**'+str(n)+':** '+models[lang].sentence_gen(20)+'\n\n'
Markdown(text)
```

Out[41]:

1: Onark oi lum tappamaaikuu, jakuonksssu tia palang jelente ma pytarytkä je hdir kenhenyr hmaiianu oolis, kanenekellesa kokuji leniensä atop otaaai.

2: Vatt kyist lakist mera keroal Menettumiedote nom vannaattiinonkan eiväännal yhde kaavaine ossil itetett ja ov simenn, kasiapainv Tuduk kaammedät meske.

3: Kukkiaalinenäjäo ainen tappuunto arvioistusv ettääkelijät luokse kanaat varaava olem luomet jukkuunlahdistammel yhämee yritt melkä messaka on vanhattim Adre mit piske.

4: Oheisesti useinänp jälji, tarvinaika Lisäksiinäy vastella mi nukuttaan ija puol, nouseens mitäähän työskena, tehnytkäännöntg oire että sitää tuolloinkää miespalvel tulkine.

5: Rve koivul ollutu päivänäki ut menettelostaa ti siltiini Isut työntek sekä: annatt rissa rinkk kertovat Kertoim Niini kirj Muk jukuritar.

6: Puhuta viel jatkunut uvat ratkai ajautu ivia vauhtiinaana bisneside lukuisiakinen jonkinlainen kuva oden katsojillesin erittäinen lman skasta, kuntopyörän kuitenkint koke.

7: Ta joissa medio kahvilanne ki Esine itava sinisenl koskaansaannostu loissa ovat toveri uusiut lapsuu epäkohta ni inhimilli yhteensä useiden edelleenj.

Conclusion

- What even is Finnish
- Very simple models can produce interesting results
- Always visualize whenever you can!

In [42]:
```
vars(john)
```

Out[42]:
```
{'background': 'Theoretical Physics',
 'company': 'KPMG',
 'github': 'johnpaton',
 'job': 'Data Science Consultant',
 'name': 'John Paton',
 'slides': 'RISE',
 'twitter': '@jd_paton'}
```

Thank you!