How many languages can a language model model?

Robert Östling

2016-12-11

Multilingual NLP

- Antiquity: one model, one language
  separate parameters, separate vocabularies

Multilingual NLP

- Antiquity: one model, one language
  separate parameters, separate vocabularies
- Recently: one model, many languages
  shared parameters, separate vocabularies

Multilingual NLP

- Antiquity: one model, one language
  separate parameters, separate vocabularies
- Recently: one model, many languages
  shared parameters, separate vocabularies
- Future: one model?

Multilingual NLP

- Antiquity: one model, one language
  separate parameters, separate vocabularies
- Recently: one model, many languages
  shared parameters, separate vocabularies
- ~~Future: one model?~~
- Dammit, Google! (Johnson et al., arXiv 2016-11-14)
  shared parameters, shared "vocabularies"

A proposition

Let's use language vectors for multilingual NLP

What do we want from language vectors?

1. Similar languages (or variants) should have similar vectors

What do we want from language vectors?

1. Similar languages (or variants) should have similar vectors
2. Vectors should be learnable from data

What do we want from language vectors?

1. Similar languages (or variants) should have similar vectors
2. Vectors should be learnable from data
3. They should encode typological parameters (word order, etc.)

What do we want from language vectors?

1. Similar languages (or variants) should have similar vectors
2. Vectors should be learnable from data
3. They should encode typological parameters (word order, etc.)
4. NLP models using the vectors should be continuous

What do we want from language vectors?

1. Similar languages (or variants) should have similar vectors
2. Vectors should be learnable from data
3. They should encode typological parameters (word order, etc.)
4. NLP models using the vectors should be continuous
   - Recall the definition of a continous function:
     $\lim_{\epsilon \to 0} f(x + \epsilon) - f(x) = 0$

What do we want from language vectors?

1. Similar languages (or variants) should have similar vectors
2. Vectors should be learnable from data
3. They should encode typological parameters (word order, etc.)
4. NLP models using the vectors should be continuous
   - Recall the definition of a continous function: $\lim_{\epsilon \to 0} f(x + \epsilon) - f(x) = 0$
   - In our interpretation: a small change in the language vector should result in small changes of our model's predictions

What do we want from language vectors?

1. Similar languages (or variants) should have similar vectors
2. Vectors should be learnable from data
3. They should encode typological parameters (word order, etc.)
4. NLP models using the vectors should be continuous
   - Recall the definition of a continous function: $\lim_{\epsilon \to 0} f(x + \epsilon) - f(x) = 0$
   - In our interpretation: a small change in the language vector should result in small changes of our model's predictions
   - $P(t_{dog} = \text{noun} | v_{eng}) = 0.9$ implies that $P(t_{dog} = \text{noun} | v_{eng} + \epsilon) \approx 0.9$

Learning vector representations

- Deep Learning^{TM} techniques are ideally suited

Learning vector representations

- Deep Learning^{TM} techniques are ideally suited
- General strategy for turning a discrete space X into vectors:

Learning vector representations

- Deep Learning^{TM} techniques are ideally suited
- General strategy for turning a discrete space X into vectors:
  1. find a supervised learning problem where X is a variable

Learning vector representations

- Deep Learning^{TM} techniques are ideally suited
- General strategy for turning a discrete space X into vectors:
  1. find a supervised learning problem where X is a variable
  2. train a neural network with a one-hot encoding for X

Learning vector representations

- Deep Learning^{TM} techniques are ideally suited
- General strategy for turning a discrete space X into vectors:
  1. find a supervised learning problem where X is a variable
  2. train a neural network with a one-hot encoding for X
  3. the input matrix now contains your X vectors

Learning vector representations

- Deep Learning^{TM} techniques are ideally suited
- General strategy for turning a discrete space X into vectors:
  1. find a supervised learning problem where X is a variable
  2. train a neural network with a one-hot encoding for X
  3. the input matrix now contains your X vectors
- Examples:

Learning vector representations

- Deep Learning™ techniques are ideally suited
- General strategy for turning a discrete space X into vectors:
  1. find a supervised learning problem where X is a variable
  2. train a neural network with a one-hot encoding for X
  3. the input matrix now contains your X vectors
- Examples:
  - Word vectors (problem: predict context word(s))

Learning vector representations

- Deep Learning^{TM} techniques are ideally suited
- General strategy for turning a discrete space X into vectors:
  1. find a supervised learning problem where X is a variable
  2. train a neural network with a one-hot encoding for X
  3. the input matrix now contains your X vectors
- Examples:
  - Word vectors (problem: predict context word(s))
  - Letter vectors (problem: predict context letter(s))

Learning vector representations

- Deep Learning™ techniques are ideally suited
- General strategy for turning a discrete space X into vectors:
  1. find a supervised learning problem where X is a variable
  2. train a neural network with a one-hot encoding for X
  3. the input matrix now contains your X vectors
- Examples:
  - Word vectors (problem: predict context word(s))
  - Letter vectors (problem: predict context letter(s))
  - Sentence vectors (problem: predict semantic equivalence)

Model

- We adapt the following tool: LSTM char-RNNLM
  = *long short-term memory character recurrent neural network language model* (phew!)

Model

- We adapt the following tool: LSTM char-RNNLM
  = *long short-term memory character recurrent neural network language model* (phew!)
- This is the E coli of neural NLP

Model

- We adapt the following tool: LSTM char-RNNLM
  = *long short-term memory character recurrent neural network language model* (phew!)
- This is the E coli of neural NLP
- Target words y_t are predicted using

$$
\begin{aligned}
P(y_t|h_t) &\propto \exp(Wh_t + b) \\
i &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
f &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
o &= \sigma(W_o x_t + U_o h_{t-1} + b_c) \\
c &= \tanh(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= fc_{t-1} + ic \\
h_t &= o \tanh(c_t)
\end{aligned}
$$

Learning language vectors

- We learn the parameters W, U, b using backpropagation

Learning language vectors

- We learn the parameters W, U, b using backpropagation
- The input x_t is concatenated from:

Learning language vectors

- We learn the parameters W, U, b using backpropagation
- The input x_t is concatenated from:
  - the character vector of y_{t-1} (last output)

Learning language vectors

- We learn the parameters W, U, b using backpropagation
- The input x_t is concatenated from:
  - the character vector of y_{t-1} (last output)
  - the language vector L of this sentence

Learning language vectors

- We learn the parameters W, U, b using backpropagation
- The input x_t is concatenated from:
  - the character vector of y_{t-1} (last output)
  - the language vector L of this sentence
- These embeddings are also learned by backpropagation

Learning language vectors

- We learn the parameters W, U, b using backpropagation
- The input x_t is concatenated from:
  - the character vector of y_{t-1} (last output)
  - the language vector L of this sentence
- These embeddings are also learned by backpropagation
- We actually have three vectors: L_1, L_2 (appanded to each LSTM) and L_3 (appended to h_t before softmax)

Learning language vectors

- We learn the parameters W, U, b using backpropagation
- The input x_t is concatenated from:
  - the character vector of y_{t-1} (last output)
  - the language vector L of this sentence
- These embeddings are also learned by backpropagation
- We actually have three vectors: L_1, L_2 (appanded to each LSTM) and L_3 (appended to h_t before softmax)
- Johnson et al. independently did essentially the same thing by adding a special language selection token in an NMT model (the main difference is that their model has to access the language vector through the attention mechanism and/or hidden LSTM states)

Data

- In principle any multilingual text collection

Data

- In principle any multilingual text collection
- I mostly use the Bible for experiments

Data

- In principle any multilingual text collection
- I mostly use the Bible for experiments
- Advantages:

Data

- In principle any multilingual text collection
- I mostly use the Bible for experiments
- Advantages:
  - same genre, (mostly) same size across languages

Data

- In principle any multilingual text collection
- I mostly use the Bible for experiments
- Advantages:
  - same genre, (mostly) same size across languages
  - easy to control for semantic similarity

Data

- In principle any multilingual text collection
- I mostly use the Bible for experiments
- Advantages:
  - same genre, (mostly) same size across languages
  - easy to control for semantic similarity
  - available in 1000+ languages

Data

- In principle any multilingual text collection
- I mostly use the Bible for experiments
- Advantages:
  - same genre, (mostly) same size across languages
  - easy to control for semantic similarity
  - available in 1000+ languages
- Disadvantages:

Data

- In principle any multilingual text collection
- I mostly use the Bible for experiments
- Advantages:
  - same genre, (mostly) same size across languages
  - easy to control for semantic similarity
  - available in 1000+ languages
- Disadvantages:
  - not much text per language (but 500M words in total)

Data

- In principle any multilingual text collection
- I mostly use the Bible for experiments
- Advantages:
  - same genre, (mostly) same size across languages
  - easy to control for semantic similarity
  - available in 1000+ languages
- Disadvantages:
  - not much text per language (but 500M words in total)
  - often a genre of its own

Data

- In principle any multilingual text collection
- I mostly use the Bible for experiments
- Advantages:
  - same genre, (mostly) same size across languages
  - easy to control for semantic similarity
  - available in 1000+ languages
- Disadvantages:
  - not much text per language (but 500M words in total)
  - often a genre of its own
  - high named entity overlap

Training

- Minibatch SGD (Adam)

Training

- Minibatch SGD (Adam)
- Different choices for sampling sentences:

Training

- Minibatch SGD (Adam)
- Different choices for sampling sentences:
  1. uniform probability over sentences

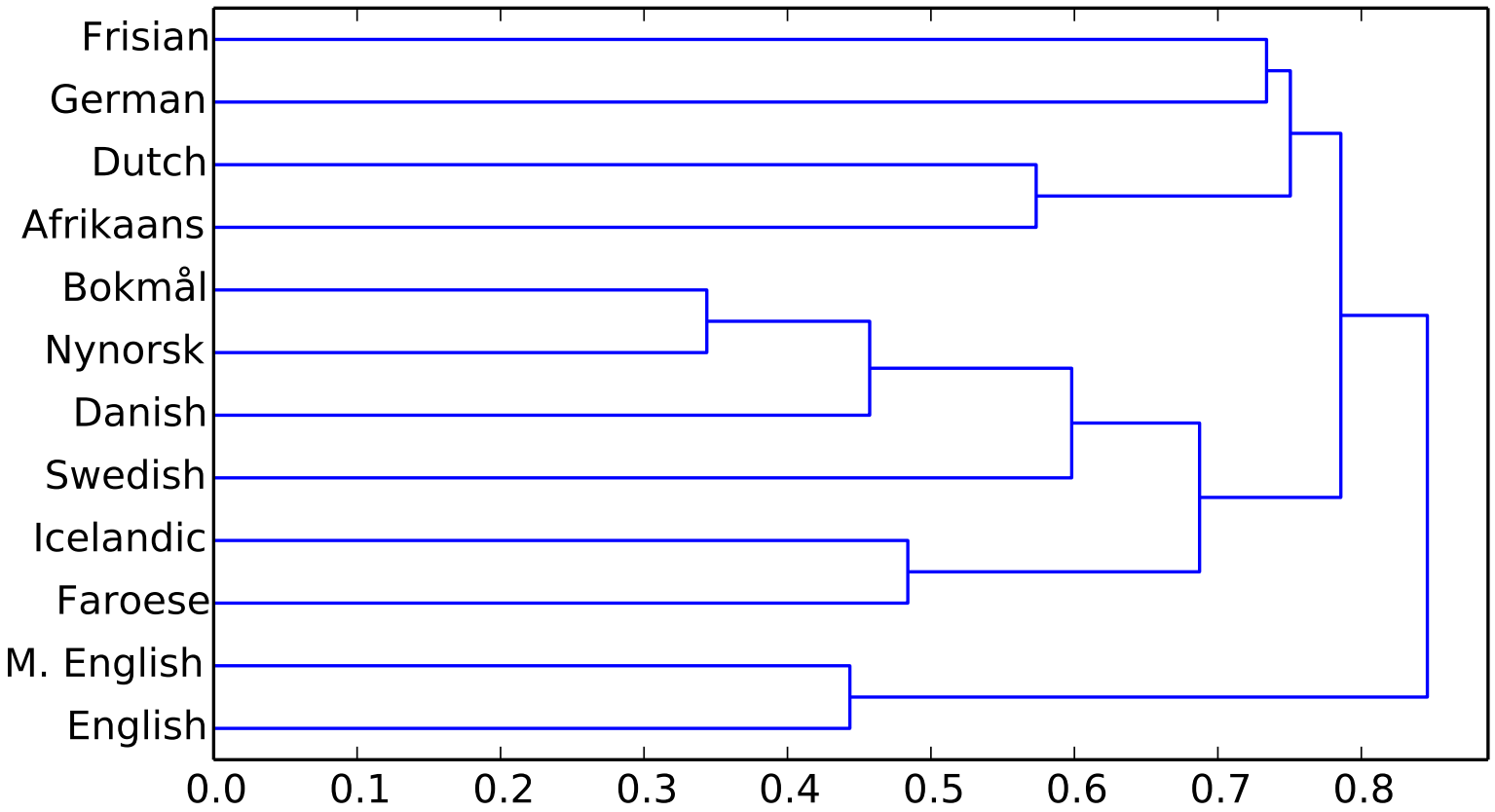
Training

- Minibatch SGD (Adam)
- Different choices for sampling sentences:
  1. uniform probability over sentences
  2. uniform probability over languages

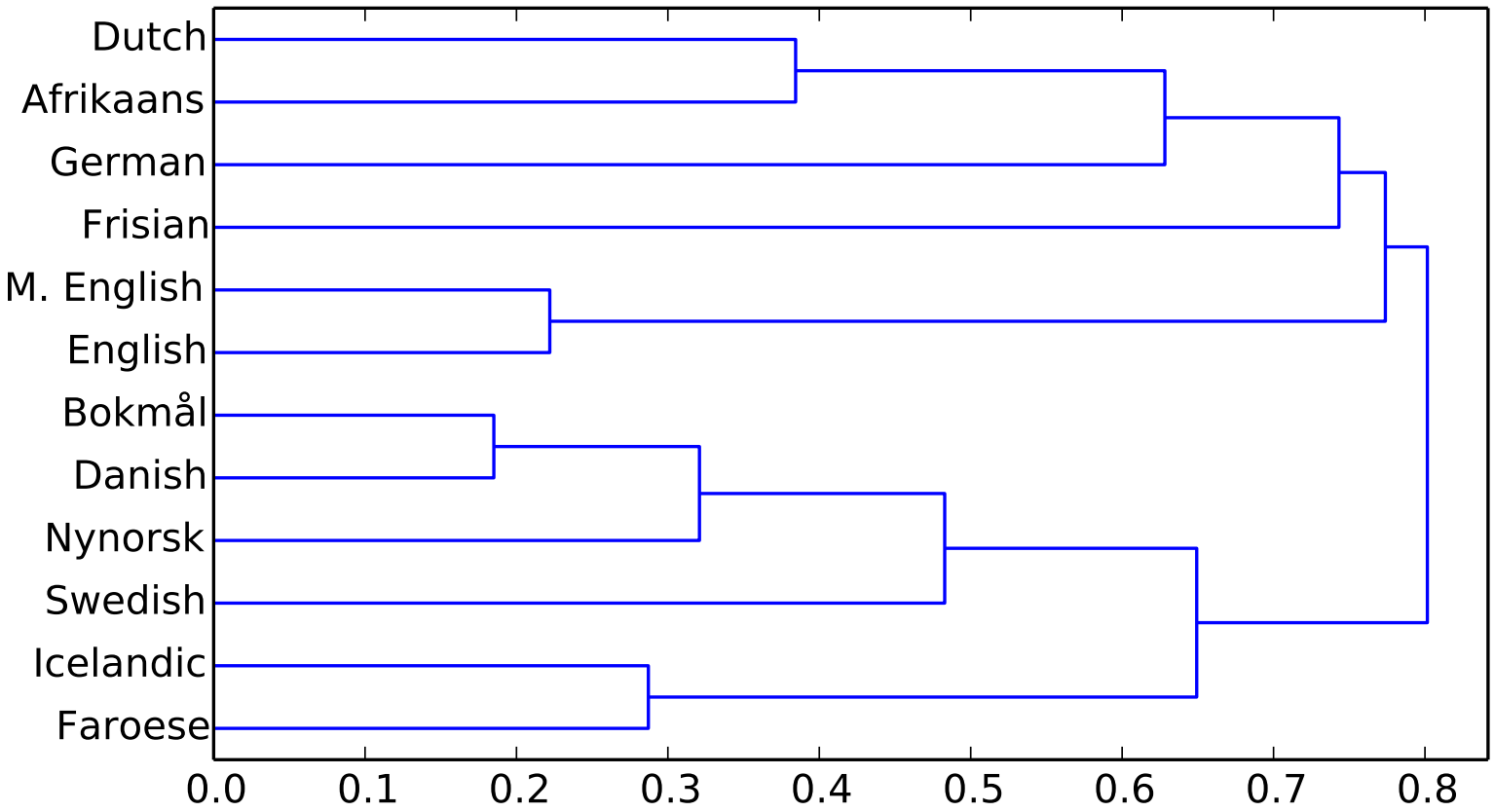
Training

- Minibatch SGD (Adam)
- Different choices for sampling sentences:
  1. uniform probability over sentences
  2. uniform probability over languages
- This takes time (several days on a GPU for large models)

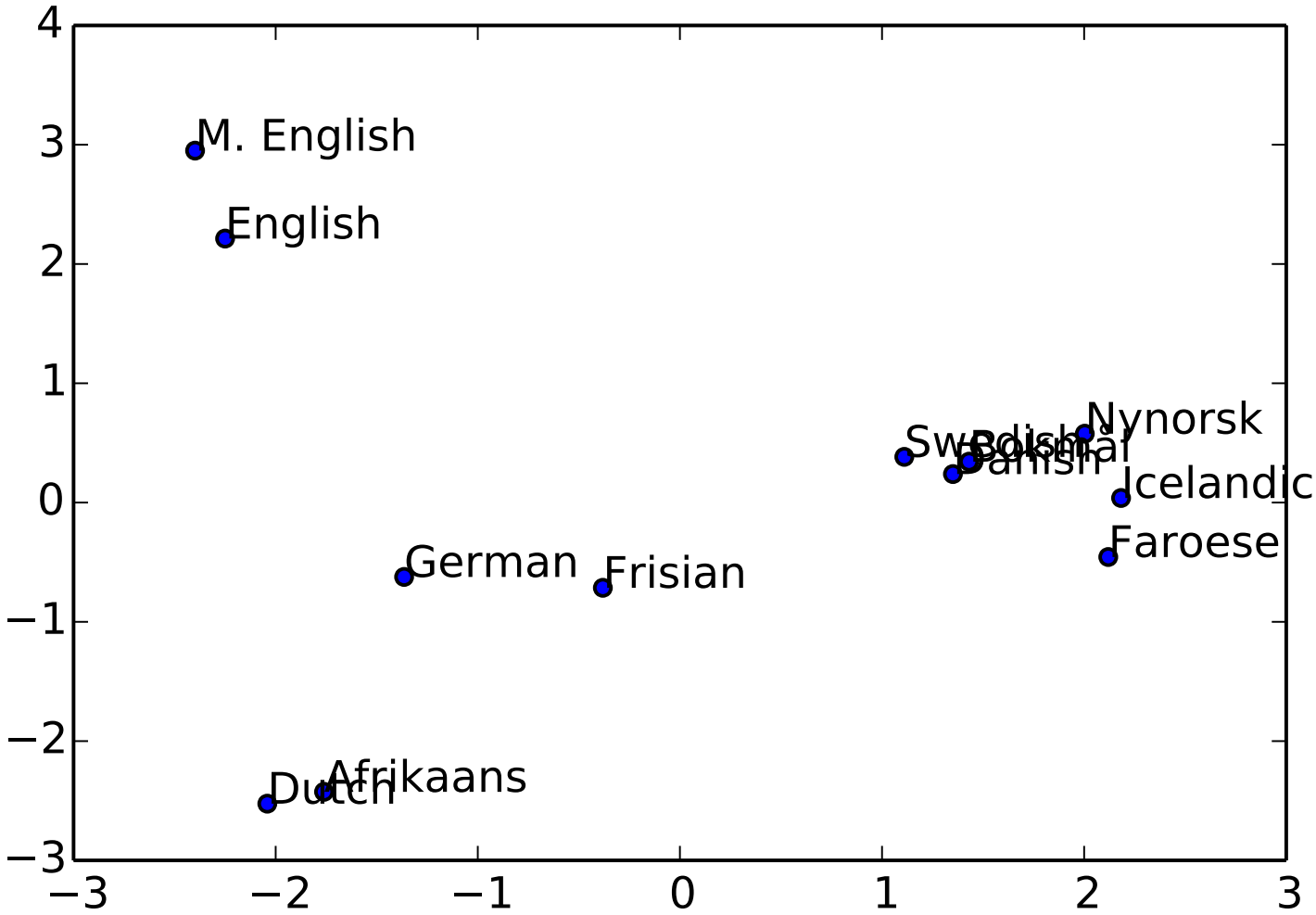
First light


Frisian
German
Dutch
Afrikaans
Bokmål
Nynorsk
Danish
Swedish
Icelandic
Faroese
M. English
English
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

Smaller model, separate initialization



Same data as before, using PCA



Capacity

- Let's look at cross-entropy of held-out Swedish verses

Capacity

- Let's look at cross-entropy of held-out Swedish verses
- All models use 1024-d LSTMs:

Capacity

- Let's look at cross-entropy of held-out Swedish verses
- All models use 1024-d LSTMs:
  - Swedish only: 1.24 bits/char

Capacity

- Let's look at cross-entropy of held-out Swedish verses
- All models use 1024-d LSTMs:
  - Swedish only: 1.24 bits/char
  - Swedish+Danish: 1.29 bits/char

Capacity

- Let's look at cross-entropy of held-out Swedish verses
- All models use 1024-d LSTMs:
  - Swedish only: 1.24 bits/char
  - Swedish+Danish: 1.29 bits/char
  - 6 scandinavian languages: 1.23 bits/char

Capacity

- Let's look at cross-entropy of held-out Swedish verses
- All models use 1024-d LSTMs:
  - Swedish only: 1.24 bits/char
  - Swedish+Danish: 1.29 bits/char
  - 6 scandinavian languages: 1.23 bits/char
  - 15 germanic languages: 1.26 bits/char

Capacity

- Let's look at cross-entropy of held-out Swedish verses
- All models use 1024-d LSTMs:
  - Swedish only: 1.24 bits/char
  - Swedish+Danish: 1.29 bits/char
  - 6 scandinavian languages: 1.23 bits/char
  - 15 germanic languages: 1.26 bits/char
  - 991 languages: 2.04 bits/char

Generating text

- We can generate from the language model

Generating text

- We can generate from the language model
- Normally we would condition on context only

Generating text

- We can generate from the language model
- Normally we would condition on context only
- In this case, we also condition on a language vector

Generating text

- We can generate from the language model
- Normally we would condition on context only
- In this case, we also condition on a language vector
- During training we learned a bunch of those, let's plug one in

Generating text

- We can generate from the language model
- Normally we would condition on context only
- In this case, we also condition on a language vector
- During training we learned a bunch of those, let's plug one in
- **English**: *on the morrow when all the people were astonied, they were afraid.*

Generating text

- We can generate from the language model
- Normally we would condition on context only
- In this case, we also condition on a language vector
- During training we learned a bunch of those, let's plug one in
- **English**: *on the morrow when all the people were astonied, they were afraid.*
- **Middle English**: *when iesus hearde yt he sayde vnto them: ye shall goe into damascus.*

Generating text

- We can generate from the language model
- Normally we would condition on context only
- In this case, we also condition on a language vector
- During training we learned a bunch of those, let's plug one in
- **English**: *on the morrow when all the people were astonied, they were afraid.*
- **Middle English**: *when iesus hearde yt he sayde vnto them: ye shall goe into damascus.*
- **German**: *und sie sprachen: du bist mein bruder.*

Generating text in non-languages

- We are not limited to existing languages

Generating text in non-languages

- We are not limited to existing languages
- Interpolating between Middle English and modern English:

Generating text in non-languages

- We are not limited to existing languages
- Interpolating between Middle English and modern English:
  - **0.30**: and thei schulen go in to alle these thingis, and schalt endure bothe in the weie

Generating text in non-languages

- We are not limited to existing languages
- Interpolating between Middle English and modern English:
  - **0.30**: and thei schulen go in to alle these thingis, and schalt endure bothe in the weie
  - **0.40**: and there was a certaine other person who was called in a dreame that he went into a mountaine.

Generating text in non-languages

- We are not limited to existing languages
- Interpolating between Middle English and modern English:
  - **0.30**: and thei schulen go in to alle these thingis, and schalt endure bothe in the weie
  - **0.40**: and there was a certaine other person who was called in a dreame that he went into a mountaine.
  - **0.44**: and the second sacrifice, and the father, and the prophet, shall be given to it.

Generating text in non-languages

- We are not limited to existing languages
- Interpolating between Middle English and modern English:
  - **0.30**: and thei schulen go in to alle these thingis, and schalt endure bothe in the weie
  - **0.40**: and there was a certaine other person who was called in a dreame that he went into a mountaine.
  - **0.44**: and the second sacrifice, and the father, and the prophet, shall be given to it.
  - **0.48**: and god sayd, i am the light of the world, and the powers of the enemies of the most high god may find first for many.

Generating text in non-languages

- We are not limited to existing languages
- Interpolating between Middle English and modern English:
  - **0.30**: and thei schulen go in to alle these thingis, and schalt endure bothe in the weie
  - **0.40**: and there was a certaine other person who was called in a dreame that he went into a mountaine.
  - **0.44**: and the second sacrifice, and the father, and the prophet, shall be given to it.
  - **0.48**: and god sayd, i am the light of the world, and the powers of the enemies of the most high god may find first for many.
  - **0.50**: but if there be some of the seruants, and to all the people, and the angels of god, and the prophets

Generating text in non-languages

- We are not limited to existing languages
- Interpolating between Middle English and modern English:
  - **0.30**: and thei schulen go in to alle these thingis, and schalt endure bothe in the weie
  - **0.40**: and there was a certaine other person who was called in a dreame that he went into a mountaine.
  - **0.44**: and the second sacrifice, and the father, and the prophet, shall be given to it.
  - **0.48**: and god sayd, i am the light of the world, and the powers of the enemies of the most high god may find first for many.
  - **0.50**: but if there be some of the seruants, and to all the people, and the angels of god, and the prophets
  - **0.52**: then he came to the gate of the city , and the bread was to be brought

Generating text in non-languages

- We are not limited to existing languages
- Interpolating between Middle English and modern English:
  - **0.30**: and thei schulen go in to alle these thingis, and schalt endure bothe in the weie
  - **0.40**: and there was a certaine other person who was called in a dreame that he went into a mountaine.
  - **0.44**: and the second sacrifice, and the father, and the prophet, shall be given to it.
  - **0.48**: and god sayd, i am the light of the world, and the powers of the enemies of the most high god may find first for many.
  - **0.50**: but if there be some of the seruants, and to all the people, and the angels of god, and the prophets
  - **0.52**: then he came to the gate of the city , and the bread was to be brought
  - **0.56**: therefore, behold, i will lose the sound of my soul, and i will not fight it into the land of egypt

Another angle

- How well does a given text match a language vector?

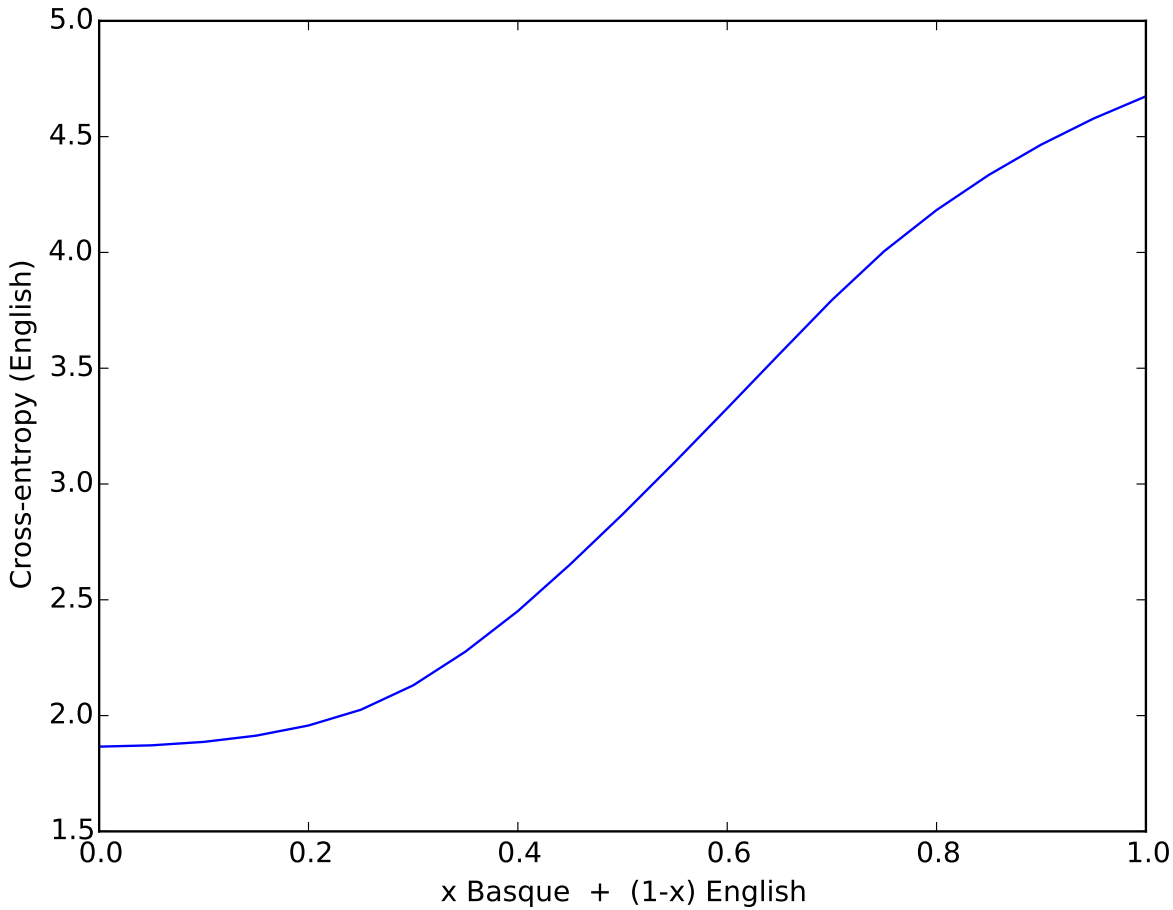
Another angle

- How well does a given text match a language vector?
- Measured by text cross-entropy

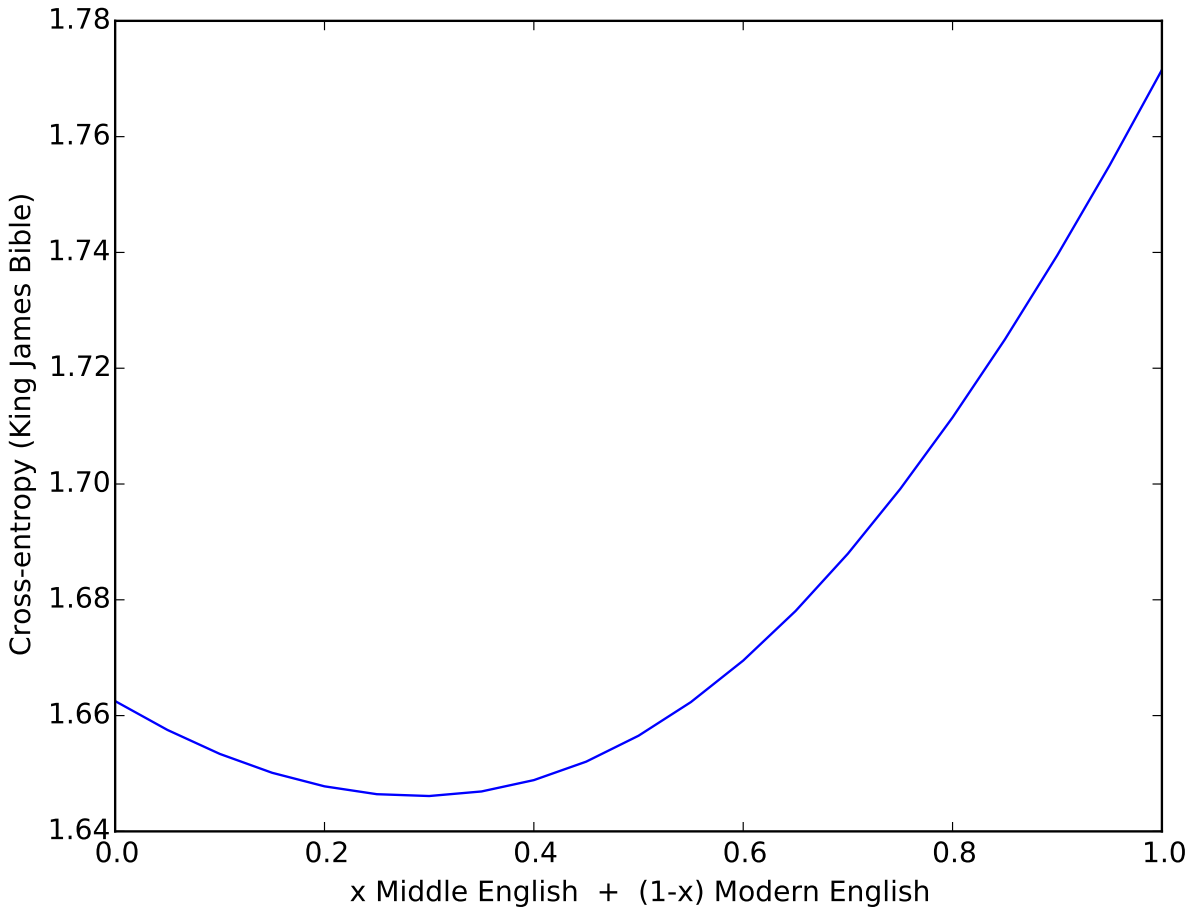
Another angle

- How well does a given text match a language vector?
- Measured by text cross-entropy
- Let's plot cross-entropy over a line in language space...

English is not Basque



Where's King James?



Finding the language

- We knew King James is related to middle and modern English

Finding the language

- We knew King James is related to middle and modern English
- What if we just have a text sample?

Finding the language

- We knew King James is related to middle and modern English
- What if we just have a text sample?
- The most obvious way is to use backpropagation:

Finding the language

- We knew King James is related to middle and modern English
- What if we just have a text sample?
- The most obvious way is to use backpropagation:
  1. Fix all parameters except the King James language vector

Finding the language

- We knew King James is related to middle and modern English
- What if we just have a text sample?
- The most obvious way is to use backpropagation:
  1. Fix all parameters except the King James language vector
  2. Optimize this vector using a (small) test sample

Finding the language

- We knew King James is related to middle and modern English
- What if we just have a text sample?
- The most obvious way is to use backpropagation:
  1. Fix all parameters except the King James language vector
  2. Optimize this vector using a (small) test sample
- A small number of sentences is enough to improve test set cross-entropy

Finding the language

- We knew King James is related to middle and modern English
- What if we just have a text sample?
- The most obvious way is to use backpropagation:
  1. Fix all parameters except the King James language vector
  2. Optimize this vector using a (small) test sample
- A small number of sentences is enough to improve test set cross-entropy
- This is an extreme form of domain/language adaptation

Finding the language

- We knew King James is related to middle and modern English
- What if we just have a text sample?
- The most obvious way is to use backpropagation:
  1. Fix all parameters except the King James language vector
  2. Optimize this vector using a (small) test sample
- A small number of sentences is enough to improve test set cross-entropy
- This is an extreme form of domain/language adaptation
- 32 training sentences brings King James test set cross-entropy from 1.128 to 1.108 (initialized with modern English)

Finding the language

- We knew King James is related to middle and modern English
- What if we just have a text sample?
- The most obvious way is to use backpropagation:
  1. Fix all parameters except the King James language vector
  2. Optimize this vector using a (small) test sample
- A small number of sentences is enough to improve test set cross-entropy
- This is an extreme form of domain/language adaptation
- 32 training sentences brings King James test set cross-entropy from 1.128 to 1.108 (initialized with modern English)
- *and he said, i will send thee forth into the fire*

Finding the language

- We knew King James is related to middle and modern English
- What if we just have a text sample?
- The most obvious way is to use backpropagation:
  1. Fix all parameters except the King James language vector
  2. Optimize this vector using a (small) test sample
- A small number of sentences is enough to improve test set cross-entropy
- This is an extreme form of domain/language adaptation
- 32 training sentences brings King James test set cross-entropy from 1.128 to 1.108 (initialized with modern English)
- *and he said, i will send thee forth into the fire*
- Note: KJV was used during training

Things we want but don't (yet) have

- We want to capture a language in a small vector

Things we want but don't (yet) have

- We want to capture a language in a small vector
- This is in general impossible, of course

Things we want but don't (yet) have

- We want to capture a language in a small vector
- This is in general impossible, of course
- But consider that WALS has 192 features

Things we want but don't (yet) have

- We want to capture a language in a small vector
- This is in general impossible, of course
- But consider that WALS has 192 features
- Exactly the number of dimensions ($3 \cdot 64$) in our language vectors! (a coincidence, I swear)

Things we want but don't (yet) have

- We want to capture a language in a small vector
- This is in general impossible, of course
- But consider that WALS has 192 features
- Exactly the number of dimensions ($3 \cdot 64$) in our language vectors! (a coincidence, I swear)
- How compactly can a language be described, if we know many other languages?

Things we want but don't (yet) have

- We want to capture a language in a small vector
- This is in general impossible, of course
- But consider that WALS has 192 features
- Exactly the number of dimensions $(3 \cdot 64)$ in our language vectors! (a coincidence, I swear)
- How compactly can a language be described, if we know many other languages?
- "Language X is basically like Y, except ..."

Things we want but don't (yet) have

- We want to capture a language in a small vector
- This is in general impossible, of course
- But consider that WALS has 192 features
- Exactly the number of dimensions ($3 \cdot 64$) in our language vectors! (a coincidence, I swear)
- How compactly can a language be described, if we know many other languages?
- "Language X is basically like Y, except ..."
  1. no final devoicing

Things we want but don't (yet) have

- We want to capture a language in a small vector
- This is in general impossible, of course
- But consider that WALS has 192 features
- Exactly the number of dimensions ($3 \cdot 64$) in our language vectors! (a coincidence, I swear)
- How compactly can a language be described, if we know many other languages?
- "Language X is basically like Y, except ..."
  1. no final devoicing
  2. its word for "calm" is borrowed from Z

Things we want but don't (yet) have

- We want to capture a language in a small vector
- This is in general impossible, of course
- But consider that WALS has 192 features
- Exactly the number of dimensions ($3 \cdot 64$) in our language vectors! (a coincidence, I swear)
- How compactly can a language be described, if we know many other languages?
- "Language X is basically like Y, except ..."
  1. no final devoicing
  2. its word for "calm" is borrowed from Z
  3. the plural suffix is *xyz*

Things we want but don't (yet) have

- We want to capture a language in a small vector
- This is in general impossible, of course
- But consider that WALS has 192 features
- Exactly the number of dimensions ($3 \cdot 64$) in our language vectors! (a coincidence, I swear)
- How compactly can a language be described, if we know many other languages?
- "Language X is basically like Y, except ..."
  1. no final devoicing
  2. its word for "calm" is borrowed from Z
  3. the plural suffix is *xyz*
  4. ...