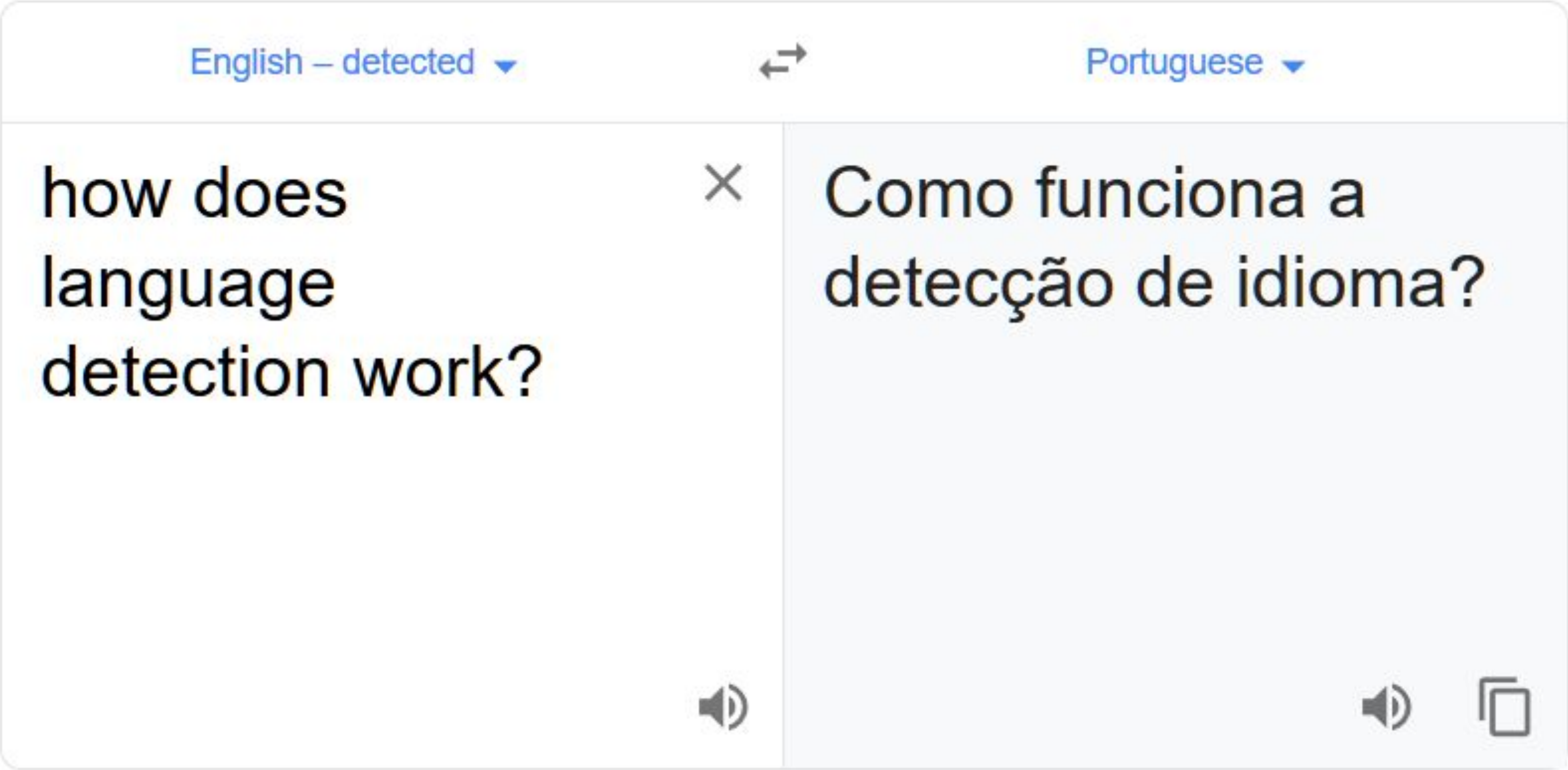
Language detection

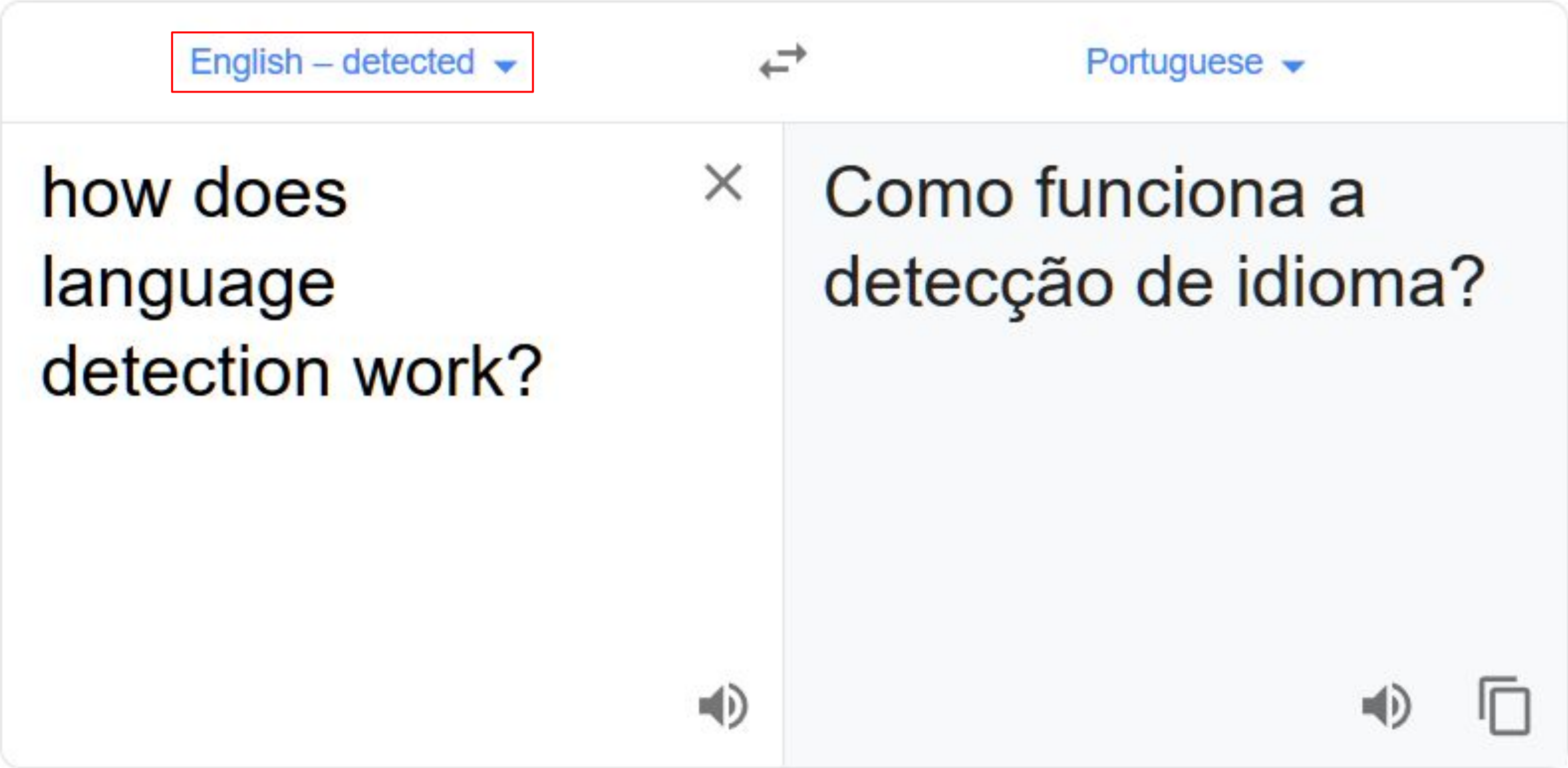
Miguel Almeida
miguelbalmeida@gmail.com

What is language detection?

What is language detection?



What is language detection?



What is language detection?

Goal: Software which goes from **text** (user input) to **language** (output)

System will learn from **examples** of what we want to do. We need to have:

- Examples of text in English
- Examples of text in Portuguese
- … and so on for all languages we want to support

Where do we get these examples?

Getting the data

- By hand:
  - Time consuming
  - Need speakers of all languages we want to support
- We can use text from Wikipedia
  - Text from **en**.wikipedia.org is ~60 GB and we know it is **English**
  - Text from **de**.wikipedia.org is ~8 GB and we know it is **German**
  - … and so on

Now we have data.

What do we do with it?

How do we use the data?

- Let’s look at sequences of 3 characters - called **trigrams**
- From ~1 MB each of English and Portuguese text:

Trigram	Count in EN text	Count in PT text	EN : PT ratio
“de “	618	10964	1 : 18
“the”	14002	320	44 : 1
“ão “	6	4458	1 : 743
“ th”	14538	338	43 : 1
“s d”	526	4074	1 : 7.7
“e f”	1508	1120	1.35 : 1

Example: “the cat”

Trigram	Count in EN text	Count in PT text
“the“	14002	320
“he ”	13030	332
“e c”	1688	1910
“ ca“	1581	2107
“cat”	429	163
PRODUCT	2.09e+17	6.97e+14

Example: “o gato”

Trigram	Count in EN text	Count in PT text
“o g“	148	594
“ ga”	581	272
“gat”	90	65
“ato“	193	433
PRODUCT	1.49e+9	4.55e+9

Language Detector v1

- At system start:
  - Grab 1 MB of text from Wikipedia for EN, PT, FR, ES, ...
  - Count frequencies of all trigrams in those files
- When user inputs text:
  - Multiply counts for all trigrams of input text
  - Language with highest product wins

Issues with v1

Issues with v1

- I have to have similar amount of text (1 MB) for all languages
- For longer sentences, product gets really big and can overflow
- Some sentences result in products of zero for all languages

What if I have different amounts of text per language?

Example: “o gato” with 10 MB of EN, 1 MB of PT

Trigram	1 MB EN	10 MB EN	1 MB PT
“o g“	148	148 x10	594
“ ga”	581	581 x10	272
“gat”	90	90 x10	65
“ato“	193	193 x10	433
PRODUCT	1.49e+9	1.49e+13	4.55e+9

Use same amount of text?

- Difficult to get decent amounts of text for minor languages

Solution: divide by total number of trigrams in file

Trigram counts:

- 1 MB English: 1001976
- 10 MB English: 10019760
- 1 MB Portuguese: 997801

Trigram	1 MB EN	10 MB EN	1 MB PT
“o g“	148/1001976	1480/10019760	594/997801
“ ga”	581/1001976	5810/10019760	272/997801
“gat”	90/1001976	900/10019760	65/997801
“ato“	193/1001976	1930/10019760	433/997801
PRODUCT	1.48e-15	1.48e-15	4.59e-15

Language Detector v2

- At system start:
  - Grab **any size** of text from Wikipedia for EN, PT, FR, ES, ...
  - Count frequencies of all trigrams **and total number of trigrams** in those files
- When user inputs text:
  - Multiply **(count / total trigrams)** for all trigrams of input text
  - Language with highest product wins

Issues with v2

Issues with v2

- ~~I have to have similar amount of text (1 MB) for all languages~~
- For longer sentences, product gets really **small** and can **underflow**
- Some sentences result in products of zero for all languages

How do I deal with underflow?

Solution: use logarithm of product instead

- If X > Y, then
  log(X) > log(Y)

- log(A*B*C*D) = log(A) + log(B) + log(C) + log(D)

Trigram	1 MB EN	1 MB PT
“o g“	log(148/1001976) = -3.83	log(594/997801) = -3.22
“ ga”	log(581/1001976) = -3.23	log(272/997801) = -3.56
“gat”	log(90/1001976) = -4.05	log(65/997801) = -4.19
“ato“	log(193/1001976) = -3.72	log(433/997801) = -3.36
sum of logs	-14.83	-14.33

Language Detector v3

- At system start:
  - Grab any size of text from Wikipedia for EN, PT, FR, ES, ...
  - Count frequencies of all trigrams and total number of trigrams in those files
- When user inputs text:
  - **Add log (count / total trigrams)** for all trigrams of input text
  - Language with highest (i.e. least negative) **sum** wins

Issues with v3

Issues with v3

- ~~I have to have similar amount of text (1 MB) for all languages~~
- ~~For longer sentences, product gets really small and can underflow~~
- ~~Some sentences result in products of zero for all languages~~
- Some sentences result in log(0), it can’t be computed

Some sentences get all products equal to zero

Example: “I stayed at Messeniuksenkatu in Finland”

Trigram	1 MB EN	1 MB PT
“i s”	log(40 / totalEN) = -4.40	log(55 / totalPT) = -4.26
“iuk”	log(0 / totalEN) = NaN	log(0 / totalPT) = NaN
“uks”	log(2 / totalEN) = -5.70	log(0 / totalPT) = NaN
“and”	log(5983 / totalEN) = -2.22	log(1686 / totalPT) = -2.77
Sum of logs	NaN	NaN

Solution (hack): add 1 to every count

Trigram	1 MB EN	1 MB PT
“i s”	log((40 + 1) / (totalEN + uEN)) = -4.39	log(55 + 1 / (totalPT + uPT)) = -4.25
“iuk”	log((0 + 1) / (totalEN + uEN)) = -6.00	log((0 + 1) / (totalPT + uPT)) = -6.00
“uks”	log((2 + 1) / (totalEN + uEN)) = -5.52	log((0 + 1) / (totalPT + uPT)) = -6.00
“and”	log((5983 + 1) / (totalEN + uEN)) = -2.22	log((1686 + 1) / (totalPT + uPT)) = -2.77
Sum of logs	-139.1	-154.6

... where uEN = unique trigrams in EN text, uPT = unique trigrams in PT text

Language Detector v4

- At system start:
  - Grab any size of text from Wikipedia for EN, PT, FR, ES, ...
  - Count frequencies of all trigrams, total number of trigrams, **and unique trigrams** in those files
- When user inputs text:
  - Add log ((count **+ 1**) / (total trigrams **+ unique trigrams**)) for all trigrams of input text
  - Language with highest (i.e. least negative) sum wins

Language Detector v4

- At system start:
  - Grab any size of text from Wikipedia for EN, PT, FR, ES, ...
  - Count frequencies of all trigrams, total number of trigrams, **and unique trigrams** in those files
- When user inputs text:
  - Add log ((count **+ 1**) / (total trigrams **+ unique trigrams**)) for all trigrams of input text
  - Language with highest (i.e. least negative) sum wins

Pseudo-code

Part 1: trigram counts and (corrected) probabilities

```
trigramCounts ← dict()
trigramProbs ← dict()
For each language L:
    trigramCounts[L] ← dict()
    totalCounts ← 0.0
    For each line in language's file:
        line ← line.lower()
        For each trigram T in that line:
            trigramCounts[L][T] ← trigramCounts[L][T] + 1.0 /* should match the values in "counts_*.txt" */
            totalCounts ← totalCounts + 1.0
    uniqueCounts ← trigramCounts[L].size() // number of distinct trigrams for language L

    trigramLogProbs[L] ← dict()
    denominator ← totalCounts + uniqueCounts
    For each trigram T in trigramCounts:
        numerator ← trigramCounts[L][T] + 1.0
        trigramLogProbs[L][T] ← log10(numerator / denominator) /* should match the values in "logprobs_*.txt" */
        trigramLogProbs[L]['__UNKNOWN__'] ← log10(1.0 / denominator)
```

Part 2: detecting the language

```
S ← input sentence
S ← S.lower()
For each language L:
    score[L] ← 0.0
    For each trigram T in S:
        If T is in trigramProbs[L]:
            score[L] ← score[L] + trigramProbs[L][T]
        Else:
            score[L] ← score[L] + trigramProbs[L]['__UNKNOWN__']
```

/* Sentence's language is the one with highest score */
/* Scores will all be negative. That's normal. :) */

Thank you!

(feel free to ask for demo)

This is a case of the Naïve Bayes algorithm

Naïve Bayes algorithm

$$\text{argmax}_l p(l|t) = \text{argmax}_l \log p(l|t) = \text{argmax}_l \frac{p(t|l)p(l))}{p(t)} = \text{argmax}_l \log p(t|l)p(l) =$$
$$\text{argmax}_l \log p(t|l) = \text{argmax}_l \log\left[p(t_1|l)p(t_2|l)...p(t_N|l)\right] = \text{argmax}_l[\log(p(t_1|l))+$$
$$\log(p(t_2|l)) + ... + \log(p(t_N|l))]$$

Maximize probability of language *l* given the input text *t*

Naïve Bayes algorithm

$$\mathrm{argmax}_l p(l|t) = \mathrm{argmax}_l \log p(l|t) = \mathrm{argmax}_l \frac{p(t|l)p(l))}{p(t)} = \mathrm{argmax}_l \log p(t|l)p(l) =$$
$$\mathrm{argmax}_l \log p(t|l) = \mathrm{argmax}_l \log [p(t_1|l)p(t_2|l)...p(t_N|l)] = \mathrm{argmax}_l [\log(p(t_1|l))+$$
$$\log(p(t_2|l)) + ... + \log(p(t_N|l))]$$

X > Y => log(X) > log(Y)

Naïve Bayes algorithm

$$\operatorname{argmax}_l p(l|t) = \operatorname{argmax}_l \log p(l|t) = \operatorname{argmax}_l \frac{p(t|l)p(l))}{p(t)} = \operatorname{argmax}_l \log p(t|l)p(l) =$$
$$\operatorname{argmax}_l \log p(t|l) = \operatorname{argmax}_l \log\left[p(t_1|l)p(t_2|l)...p(t_N|l)\right] = \operatorname{argmax}_l [\log(p(t_1|l))+$$
$$\log(p(t_2|l)) + ... + \log(p(t_N|l))]$$

p(A|B) = p(B|A) p(A) / p(B)

(Bayes' Theorem)

Naïve Bayes algorithm

$$\text{argmax}_l p(l|t) = \text{argmax}_l \log p(l|t) = \text{argmax}_l \frac{p(t|l)p(l))}{p(t)} = \text{argmax}_l \log p(t|l)p(l) =$$
$$\text{argmax}_l \log p(t|l) = \text{argmax}_l \log \left[p(t_1|l)p(t_2|l)...p(t_N|l)\right] = \text{argmax}_l [\log(p(t_1|l))+$$
$$\log(p(t_2|l)) + ... + \log(p(t_N|l))]$$

p(t) does not depend on l

Naïve Bayes algorithm

$$\text{argmax}_l p(l|t) = \text{argmax}_l \log p(l|t) = \text{argmax}_l \frac{p(t|l)p(l))}{p(t)} = \text{argmax}_l \log p(t|l)p(l) =$$
$$\text{argmax}_l \log p(t|l) = \text{argmax}_l \log [p(t_1|l)p(t_2|l)...p(t_N|l)] = \text{argmax}_l [\log(p(t_1|l)) +$$
$$\log(p(t_2|l)) + ... + \log(p(t_N|l))]$$

First assumption

Probability of the language of the user input is the same for all languages

(called Uniform Prior)

Naïve Bayes algorithm

$$\mathrm{argmax}_l p(l|t) = \mathrm{argmax}_l \log p(l|t) = \mathrm{argmax}_l \frac{p(t|l)p(l))}{p(t)} = \mathrm{argmax}_l \log p(t|l)p(l) =$$
$$\mathrm{argmax}_l \log p(t|l) = \mathrm{argmax}_l \log\,[p(t_1|l)p(t_2|l)...p(t_N|l)] = \mathrm{argmax}_l[\log(p(t_1|l))+$$
$$\log(p(t_2|l)) + ... + \log(p(t_N|l))]$$

Second assumption

Probability of the text is equal to the product of the probabilities of its trigrams

(that's why it's called **Naïve** Bayes)

Naïve Bayes algorithm

$$\text{argmax}_l p(l|t) = \text{argmax}_l \log p(l|t) = \text{argmax}_l \frac{p(t|l)p(l))}{p(t)} = \text{argmax}_l \log p(t|l)p(l) =$$
$$\text{argmax}_l \log p(t|l) = \text{argmax}_l \log\left[p(t_1|l)p(t_2|l)...p(t_N|l)\right] = \text{argmax}_l[\log(p(t_1|l))+$$
$$\log(p(t_2|l)) + ... + \log(p(t_N|l))]$$

log(A*B) = log(A) + log(B)

Example for French

Example: “the cat”

Trigram	Count in EN text	Count in FR text	Count in PT text
“the“	14002	380	320
“he ”	13030	547	332
“e c”	1688	2725	1910
“ ca“	1581	1372	2107
“cat”	429	322	163
PRODUCT	2.09e+17	2.50e+15	6.97e+14

Example: “le chat”

Trigram	Count in EN text	Count in FR text	Count in PT text
“le “	1329	6995	531
“e c”	1688	2725	1910
“ ch”	1331	1270	635
“cha“	1058	1007	393
“hat”	988	22	18
PRODUCT	3.12e+15	5.36e+14	4.56e+12