COMPUTER SCIENCE + LANGUAGES = ARTIFICIAL INTELLIGENCE

Pamela Toman

13 November 2016

Do you recognize the language?

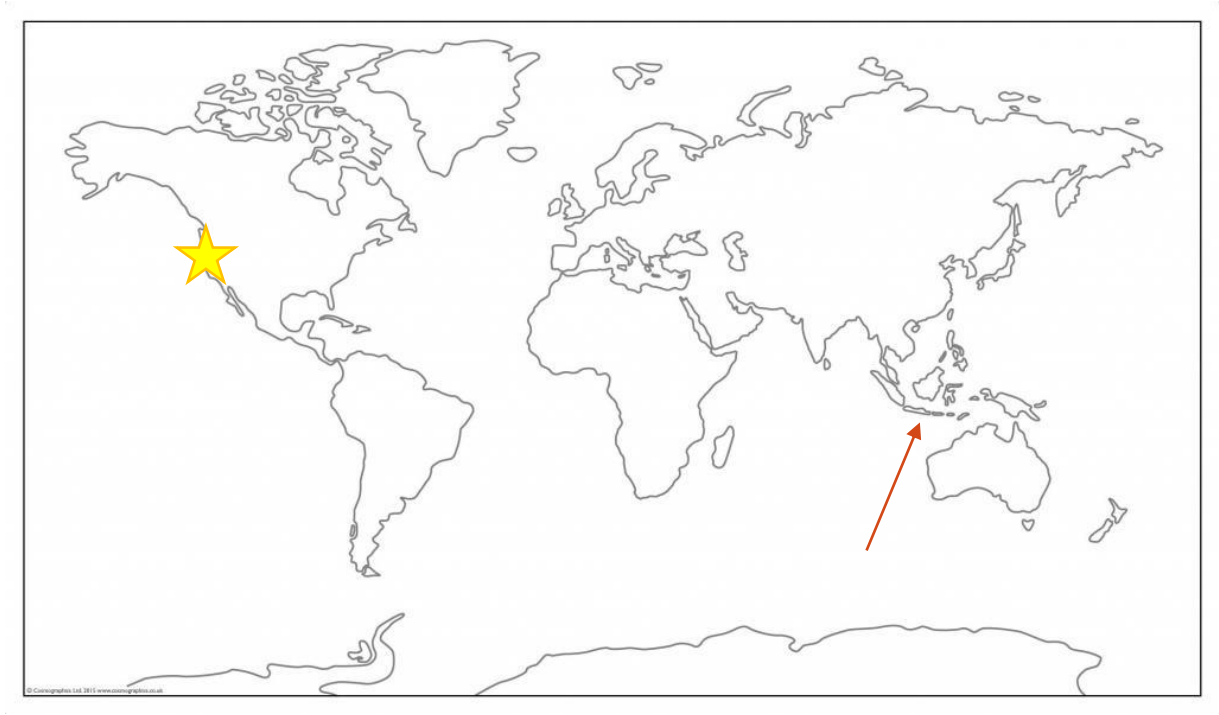
Ring sajeroning pangawedar puniki tan dados katarka tur nguwehin inggian negara, paguyuban, utawi silih tunggal janma, …	Rangx lol ntuad nad nit renf hox tiaox venx, zhit zhund caik uat uat dab yus deb njoux dout renf hox ib lob gox juab, …	Akukho lutho okukuloluGunyazo okuyohunyushwa ngokuthi kuthi uMbuso mumbe, …

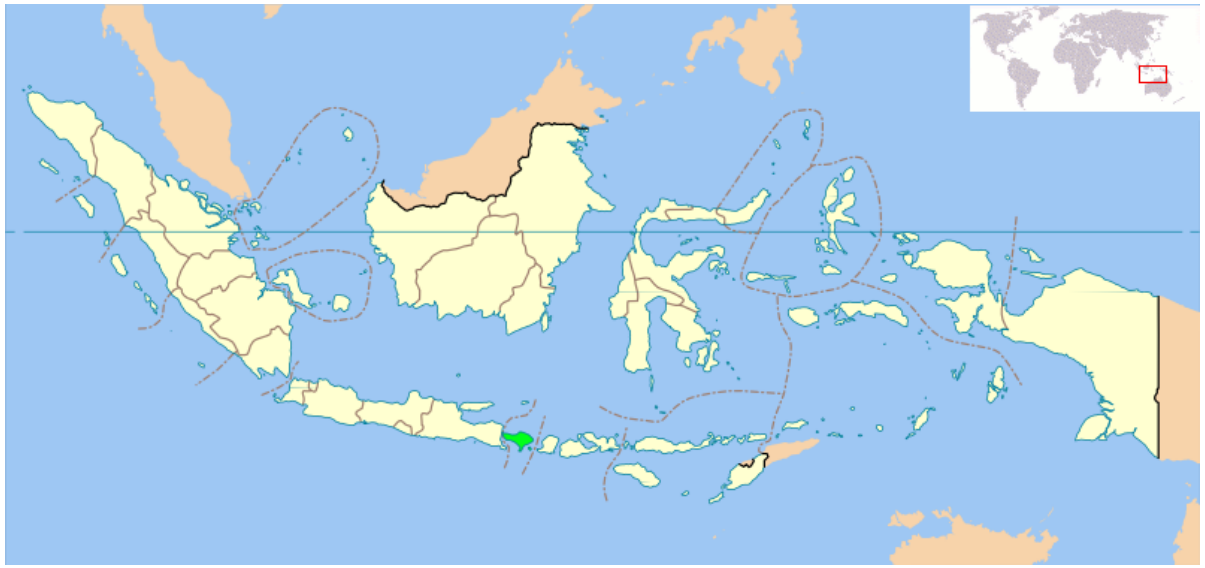
Do you recognize the language?

Balinese	Hmong	Zulu
Ring sajeroning pangawedar puniki tan dados katarka tur nguwehin inggian negara, paguyuban, utawi silih tunggal janma, …	Rangx lol ntuad nad nit renf hox tiaox venx, zhit zhund caik uat uat dab yus deb njoux dout renf hox ib lob gox juab, …	Akukho lutho okukuloluGunyazo okuyohunyushwa ngokuthi kuthi uMbuso mumbe, …

Balinese

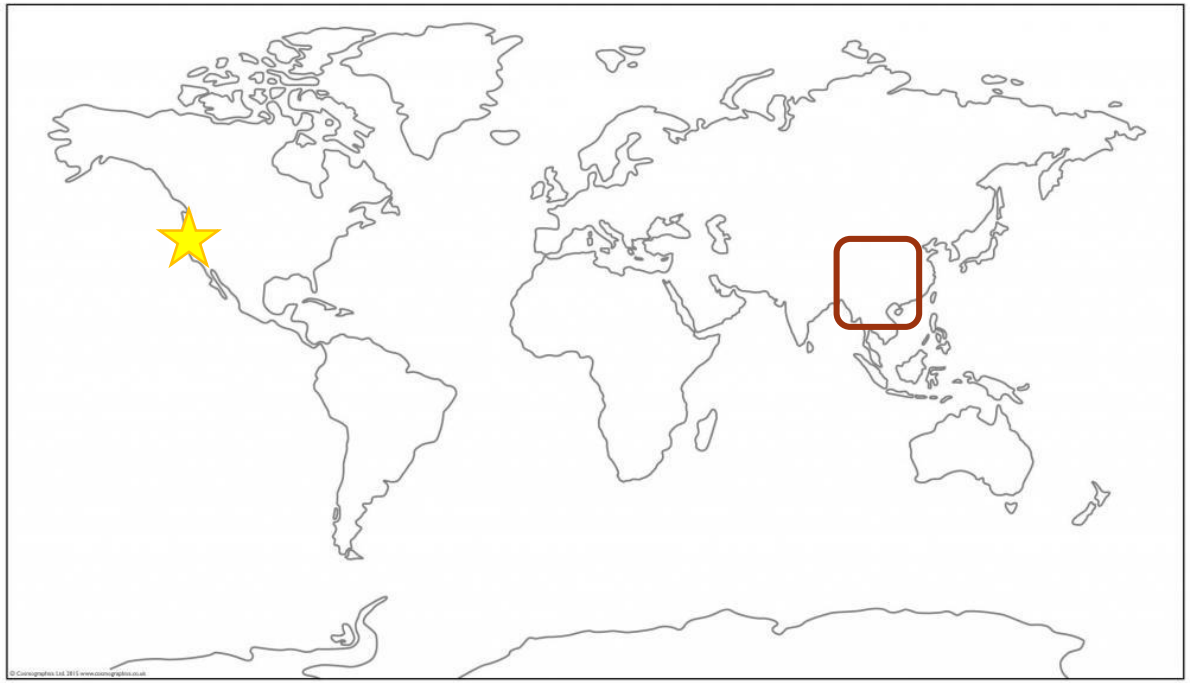
- Spoken in Indonesia (Asia/Oceania)
  - On the island of Bali
- 3.3 million speakers

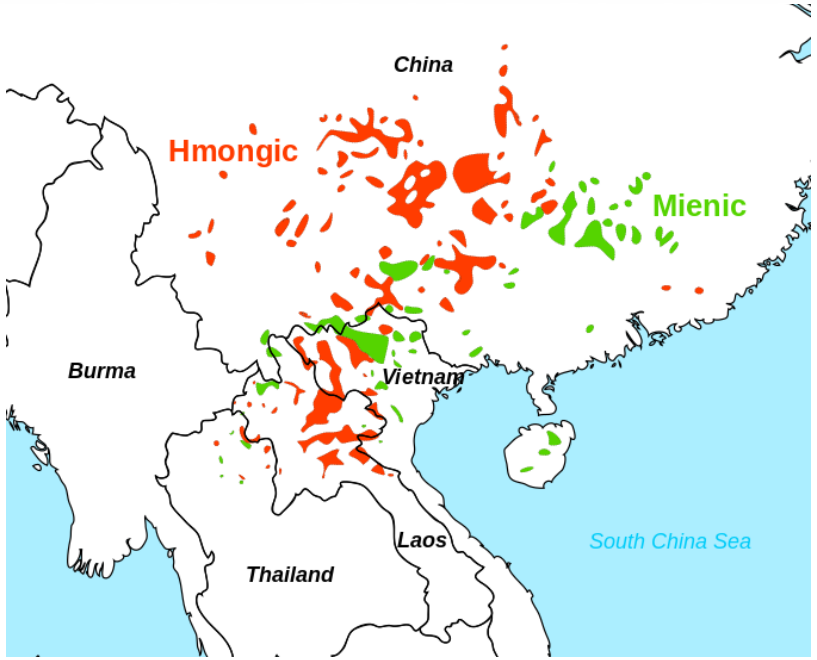




Hmong

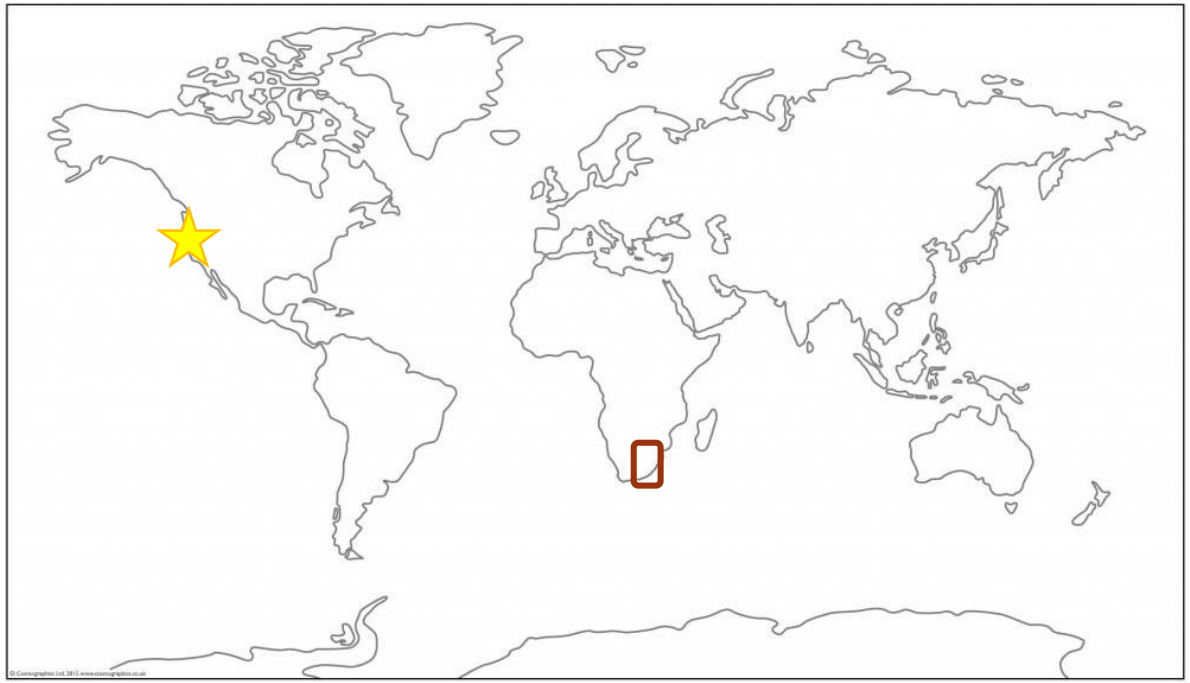
- Spoken in southeast Asia (Asia)
  - In China, Vietnam, Thailand, Laos
- 2.7 million speakers

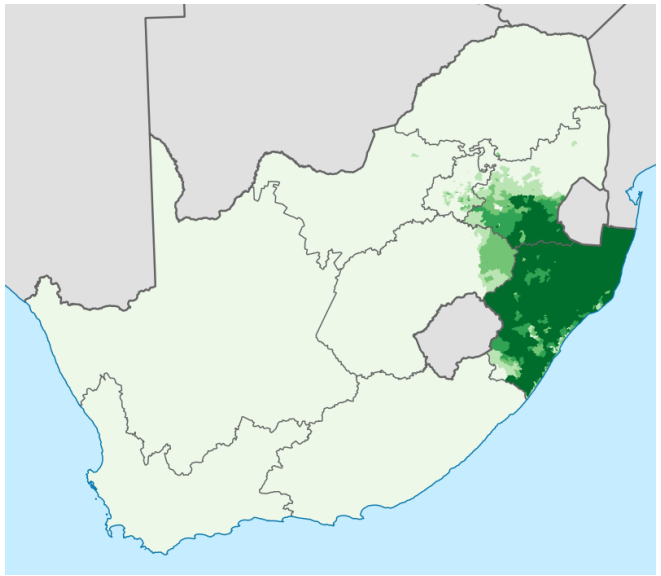




Zulu

- Spoken in South Africa (Africa)
  - On the eastern side
- 10 million speakers





Can you guess this language?

Lamalungelo nenkululeko akunakusetshenziswa ngokuphambana nezinhloso nemigomo yeNhlangano yeZizwe.

?	?	?
Balinese	Hmong	Zulu
Ring sajeroning pangawedar puniki tan dados katarka tur nguwehin inggian negara, paguyuban, utawi silih tunggal janma, hak nyarengin saluir pekaryaan, utawi ngelaksanayang saparisolah sane matujon ngerusak hak-hak lan kebebasan-kebebasan sane mungguh ring sajeroning pangawedar puniki.	Rangx lol ntuad nad nit renf hox tiaox venx, zhit zhund caik uat uat dab yus deb njoux dout renf hox ib lob gox juab, renf hox nbaox buab beus las shik renf hox ib dol nenb muax ndos uat renf hox yat zhof uat bual rangx lol ntuad nad jif muax nit rent hox ndas dos tab zif youx nit hox dongf hox zhed deut del.	Akukho lutho okukuloluGunyazo okuyohunyushwa ngokuthi kuthi uMbuso mumbe, iqembu noma umuntu unelungelo lokwenza noma isiphi isenzo sokuthikameza noma imaphi amalungebo nenkululeko echazwe ngaphezulu.

Did you do it like this?

- Matching words?
- Interesting combinations of letters?
- Capital letters in the middle of words?

Lamalungelo nenkululeko akunakusetshenziswa ngokuphambana nezinhloso nemigomo yeNhlangano yeZizwe.

Balinese	Hmong	Zulu
Ring sajeroning pangawedar puniki tan dados katarka tur nguwehin inggian negara, paguyuban, utawi silih tunggal janma, hak nyarengin saluir pekaryaan, utawi ngelaksanayang saparisolah sane matujon ngerusak hak-hak lan kebebasan-kebebasan sane mungguh ring sajeroning pangawedar puniki.	Rangx lol ntuad nad nit renf hox tiaox venx, zhit zhund caik uat uat dab yus deb njoux dout renf hox ib lob gox juab, renf hox nbaox buab beus las shik renf hox ib dol nenb muax ndos uat renf hox yat zhof uat bual rangx lol ntuad nad jif muax nit rent hox ndas dos tab zif youx nit hox dongf hox zhed deut del.	Akukho lutho okukuloluGunyazo okuyohunyushwa ngokuthi kuthi uMbuso mumbe, iqembu noma umuntu unelungelo lokwenza noma isiphi isenzo sokuthikameza noma imaphi amalungebo nenkululeko echazwe ngaphezulu.

How about this one?

Zaox dous shid yongf dex ndas dos tab zif youx nad, zhit guangd nyaob zhangd qinx kuangf dus, …

?	?	?
Balinese	Hmong	Zulu
Ring sajeroning pangawedar puniki tan dados katarka tur nguwehin inggian negara, paguyuban, utawi silih tunggal janma, hak nyarengin saluir pekaryaan, utawi ngelaksanayang saparisolah sane matujon ngerusak hak-hak lan kebebasan-kebebasan sane mungguh ring sajeroning pangawedar puniki.	Rangx lol ntuad nad nit renf hox tiaox venx, zhit zhund caik uat uat dab yus deb njoux dout renf hox ib lob gox juab, renf hox nbaox buab beus las shik renf hox ib dol nenb muax ndos uat renf hox yat zhof uat bual rangx lol ntuad nad jif muax nit rent hox ndas dos tab zif youx nit hox dongf hox zhed deut del.	Akukho lutho okukuloluGunyazo okuyohunyushwa ngokuthi kuthi uMbuso mumbe, iqembu noma umuntu unelungelo lokwenza noma isiphi isenzo sokuthikameza noma imaphi amalungebo nenkululeko echazwe ngaphezulu.

What might you count to identify a language?

(We need numbers because computers need numbers!)

The frequencies of letters help

Balinese

Ring sajeroning …

Counts of letters

R ?

The frequencies of letters help

Balinese

Ring sajeroning …

Counts of letters

R l
i ?

The frequencies of letters help

Balinese

Ring sajeroning …

Counts of letters

```
R  1
i  2
n  ?
```

The frequencies of letters help

Balinese

Ring sajeroning …

Counts of letters

```
R 1
i 2
n 3
```

The frequencies of letters help

Balinese

Ring sajeroning …

Counts of letters

```
R  1
i  2
n  3
 …
```

The frequencies of letters help

Balinese

Ring sajeroning …

Counts of letters

```
R  1
i  2
n  3
…
```

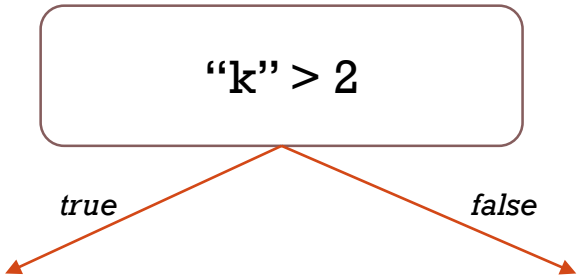
We call these 1-letter sequences "*unigrams*"

Machines can "learn" how to identify languages too

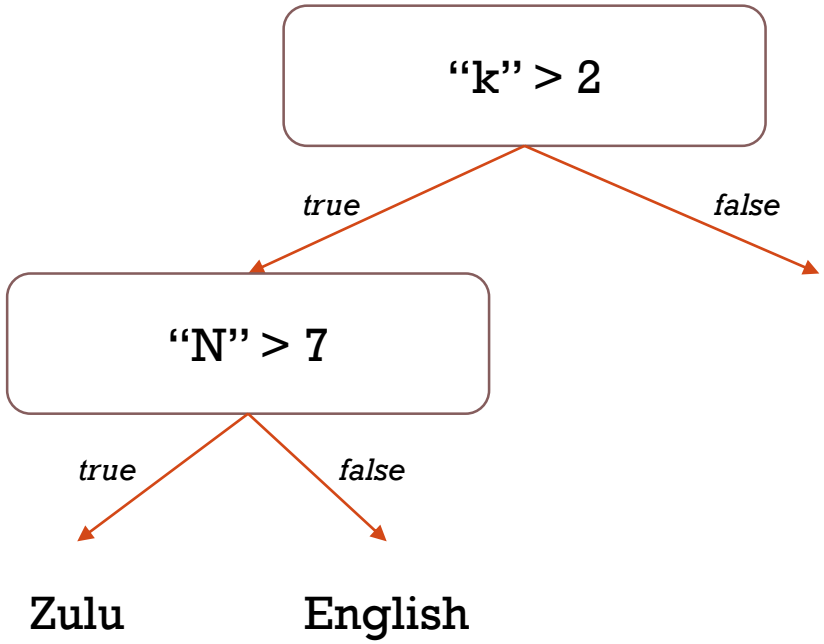
- The computer is given a set of *input variables* + a *target variable*
  - *One example:* Counts of letters in a text (input variables)
    Language name (target variable)
  - *Another example*: Gender, age, uses computer daily (input variables)
    Likes video games? (target variable)
- Then we give it a new set of variables, and we ask it to predict the target
  - "I have this new text someone just typed into Google Translate.
    What language is it (so I can correctly translate it for them)?'
  - "I have a female 55-year-old who uses the computer daily.
    Does she like video games?"

How do we predict the target variable?!?

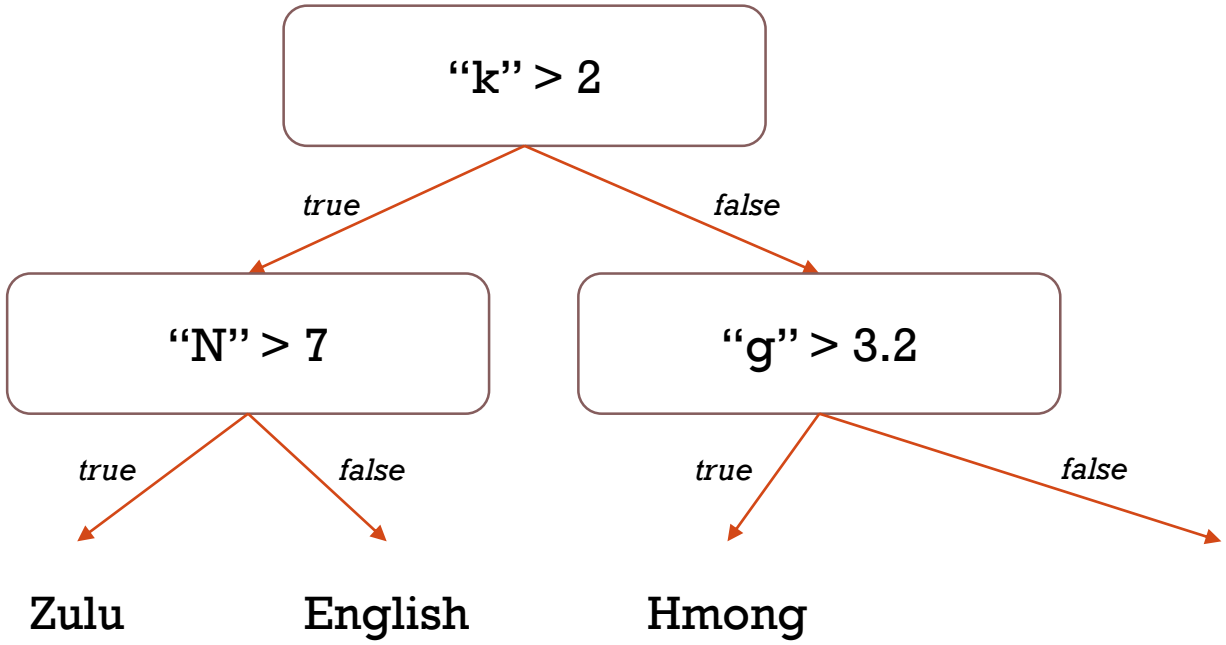
One way to predict is with a "*decision tree*"



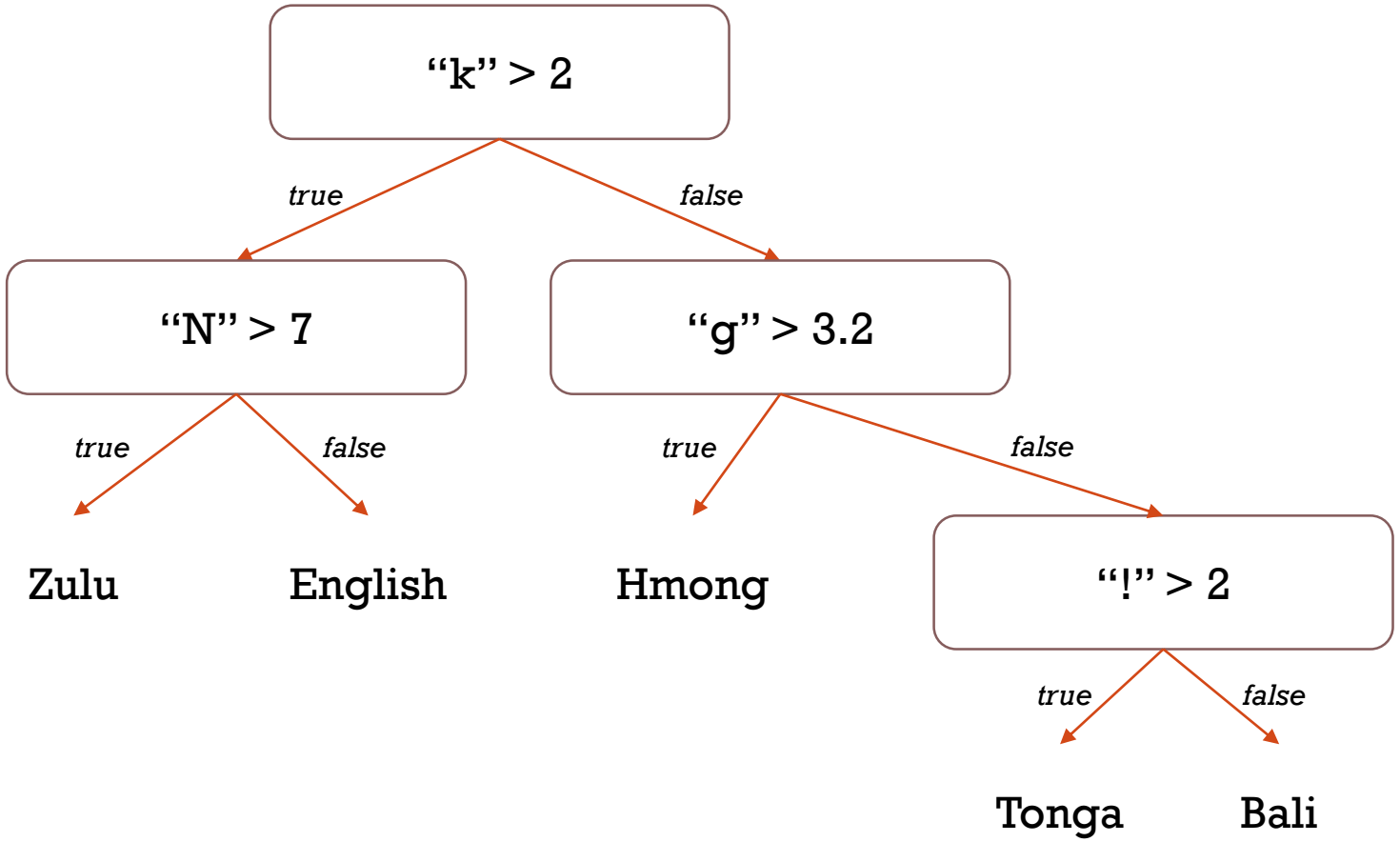
One way to predict is with a "*decision tree*"



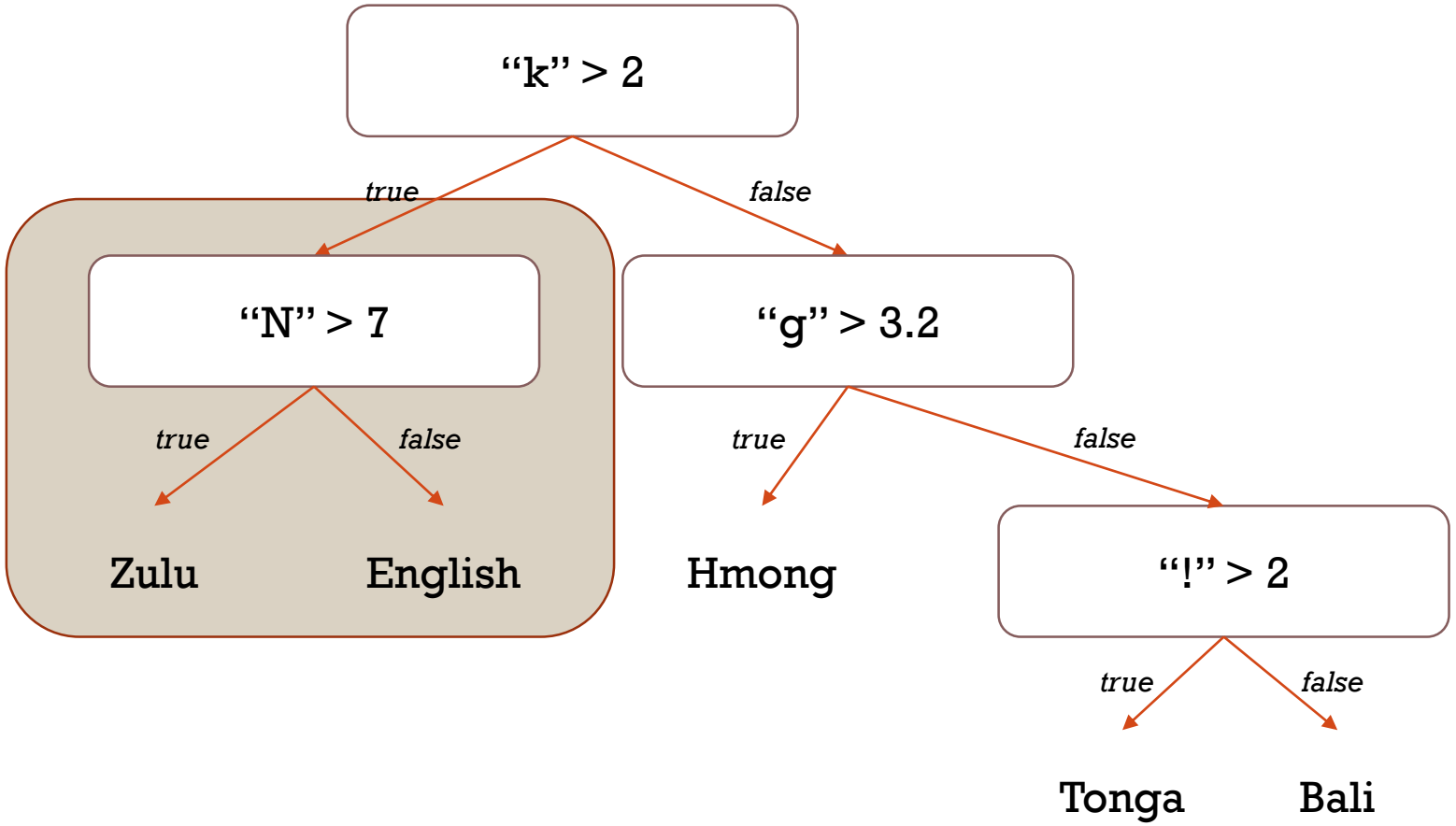
One way to predict is with a "*decision tree*"



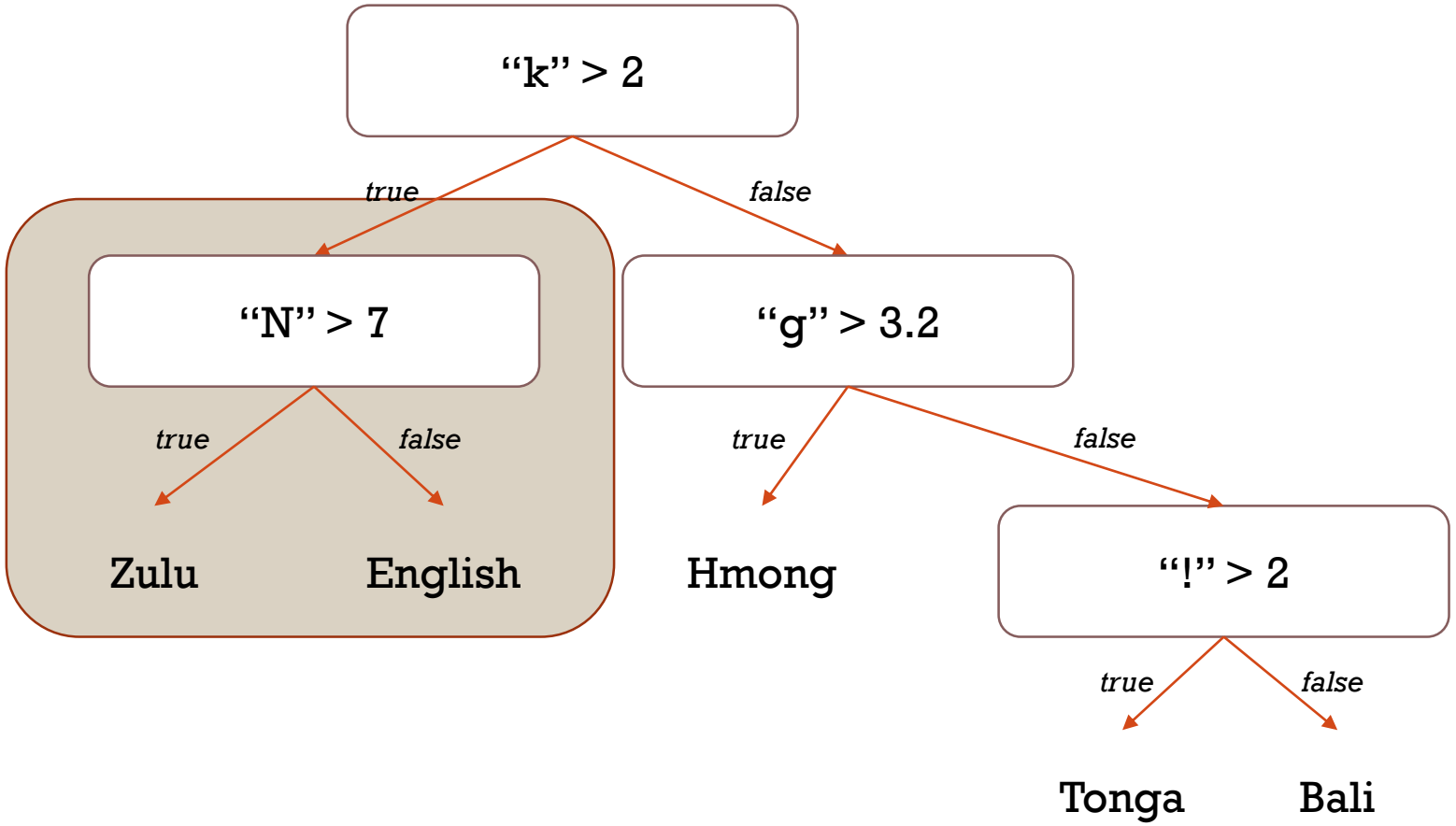
One way to predict is with a "*decision tree*"



One way to predict is with a "*decision tree*"

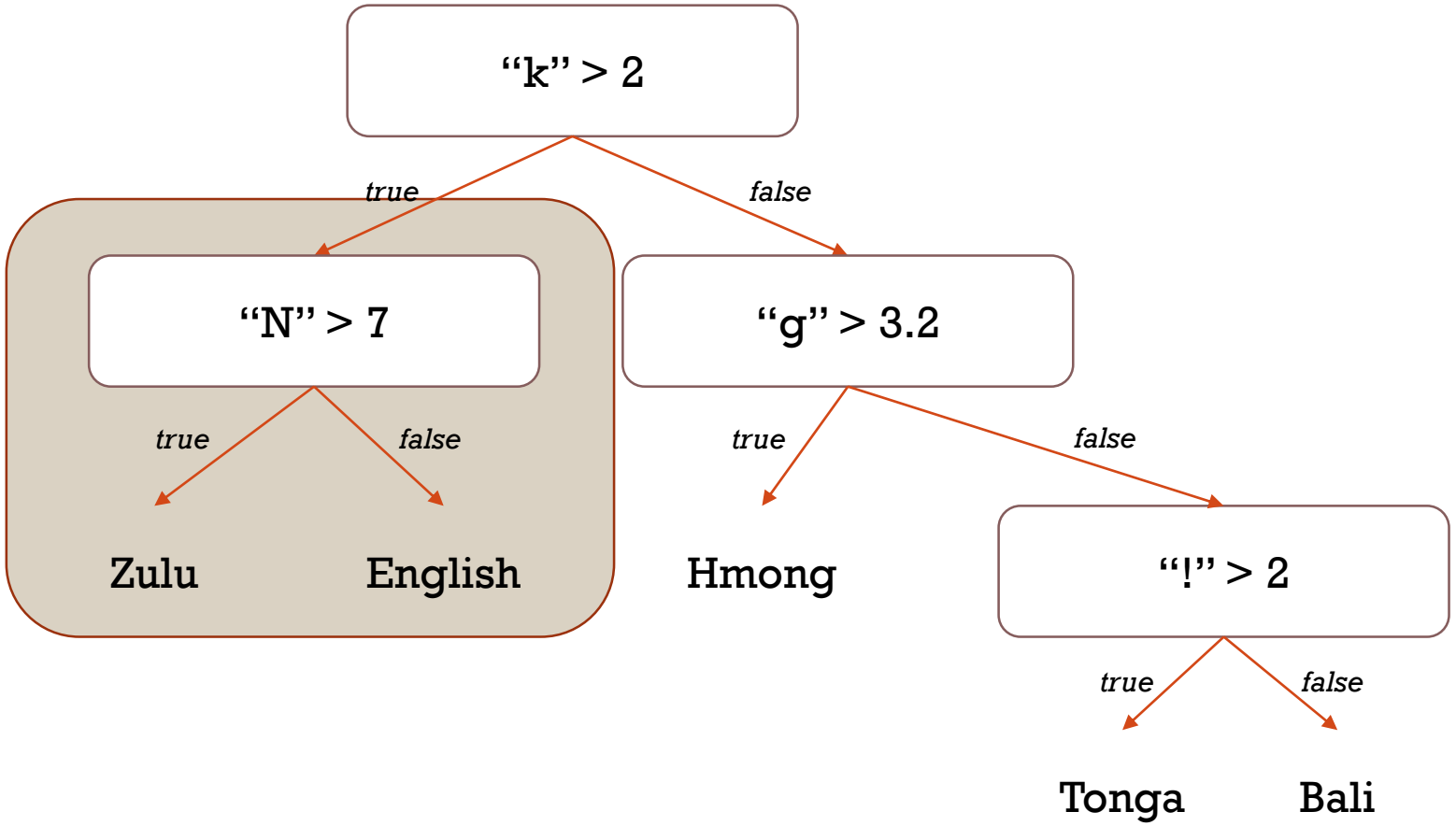


One way to predict is with a "*decision tree*"



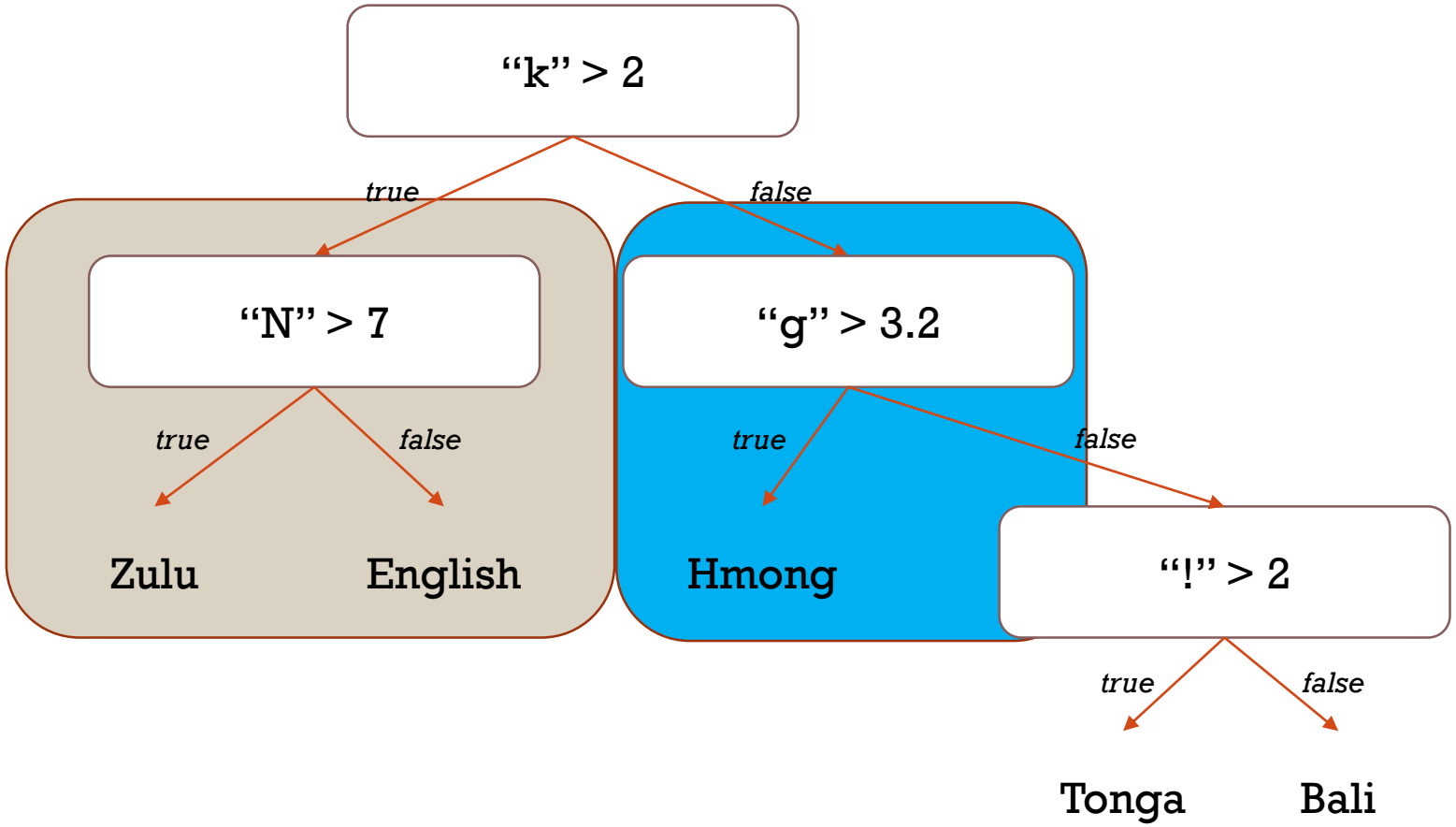

```
if count["N"] > 7:
    return "Zulu"
```

One way to predict is with a "*decision tree*"



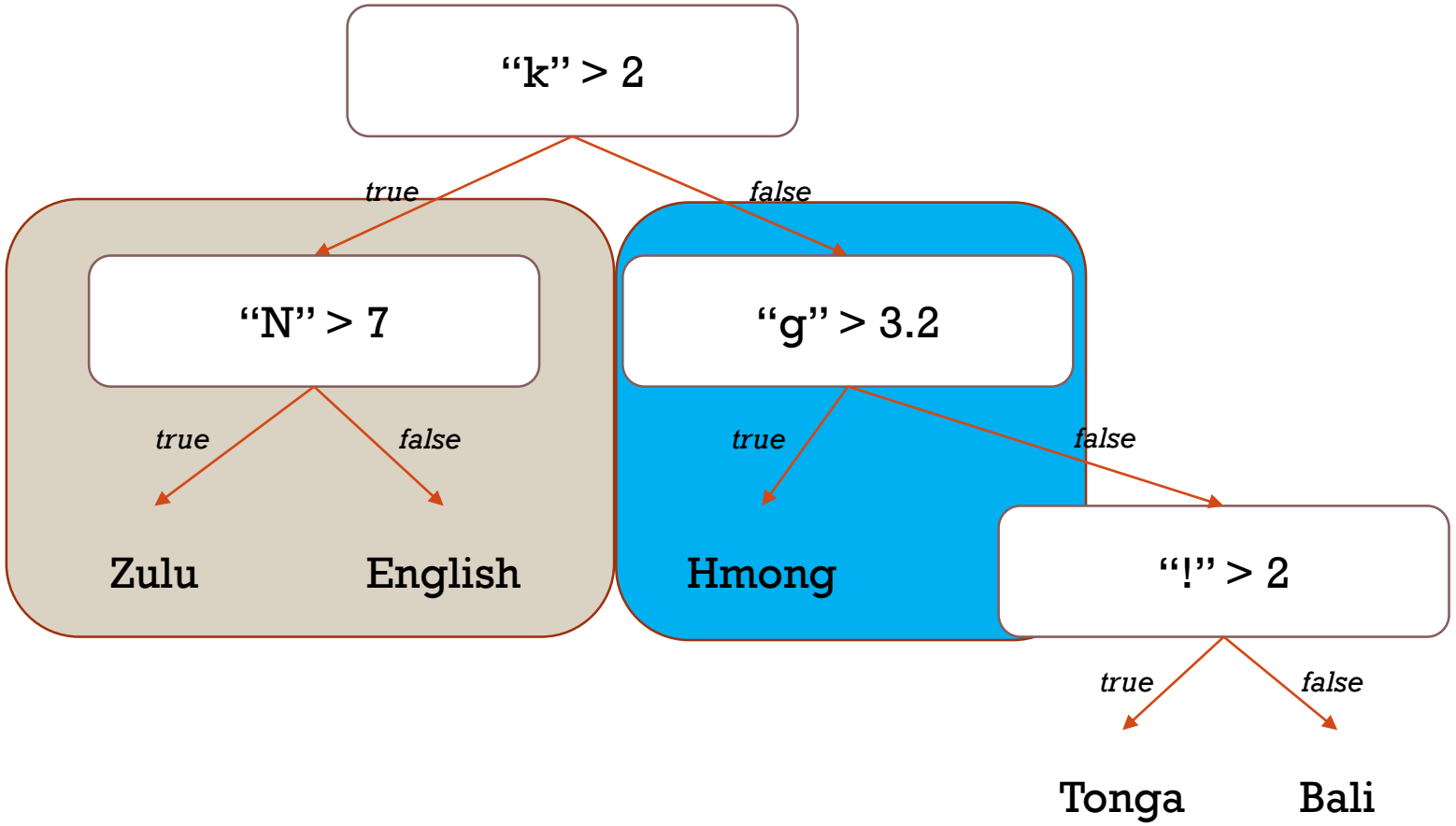

```
if count["N"] > 7:
    return "Zulu"

else:
    return "English"
```

One way to predict is with a "*decision tree*"




```
if count["N"] > 7:
    return "Zulu"

else:
    return "English"
```

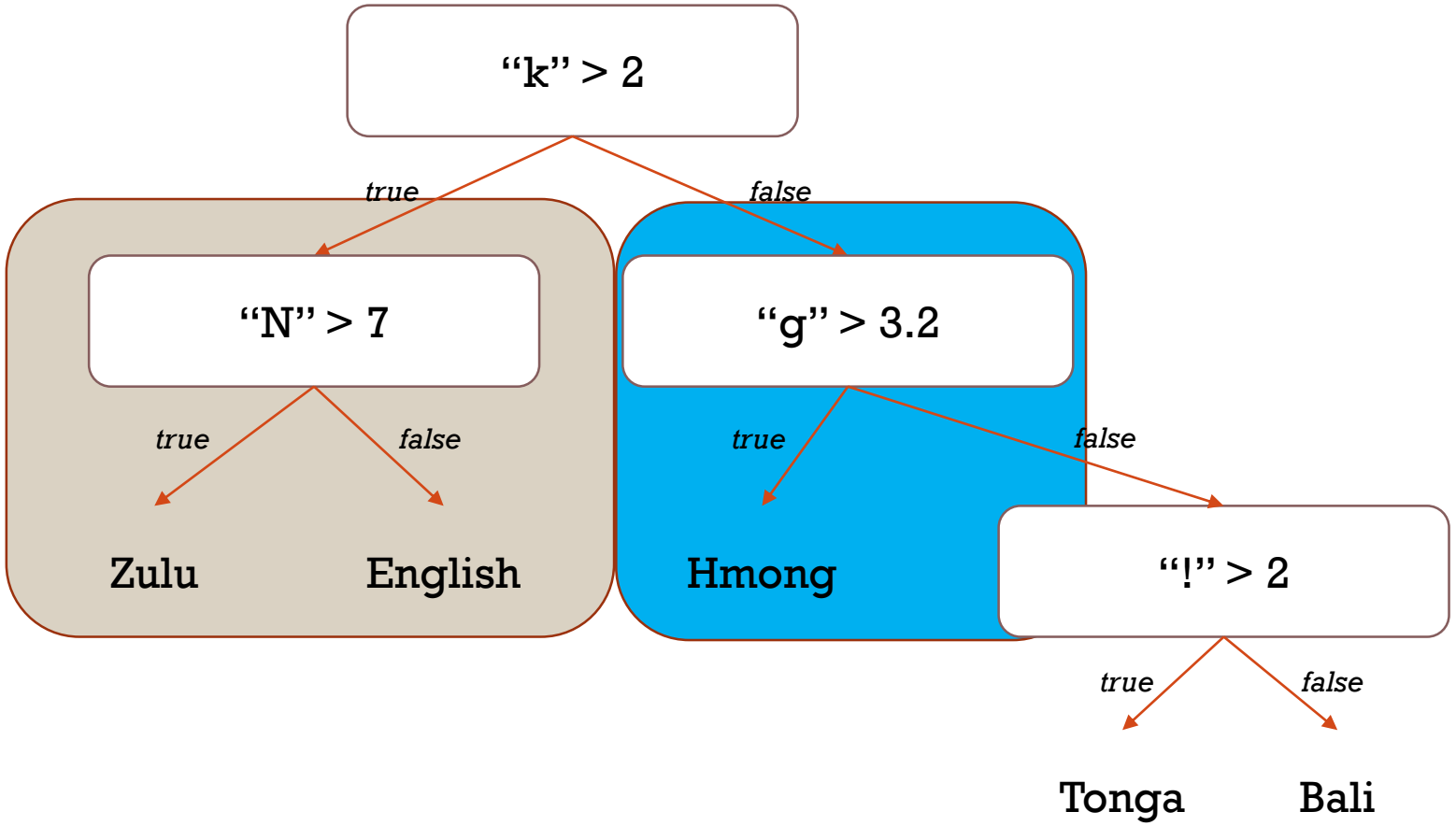
One way to predict is with a "*decision tree*"




```
if count["N"] > 7:
    return "Zulu"
else:
    return "English"
```

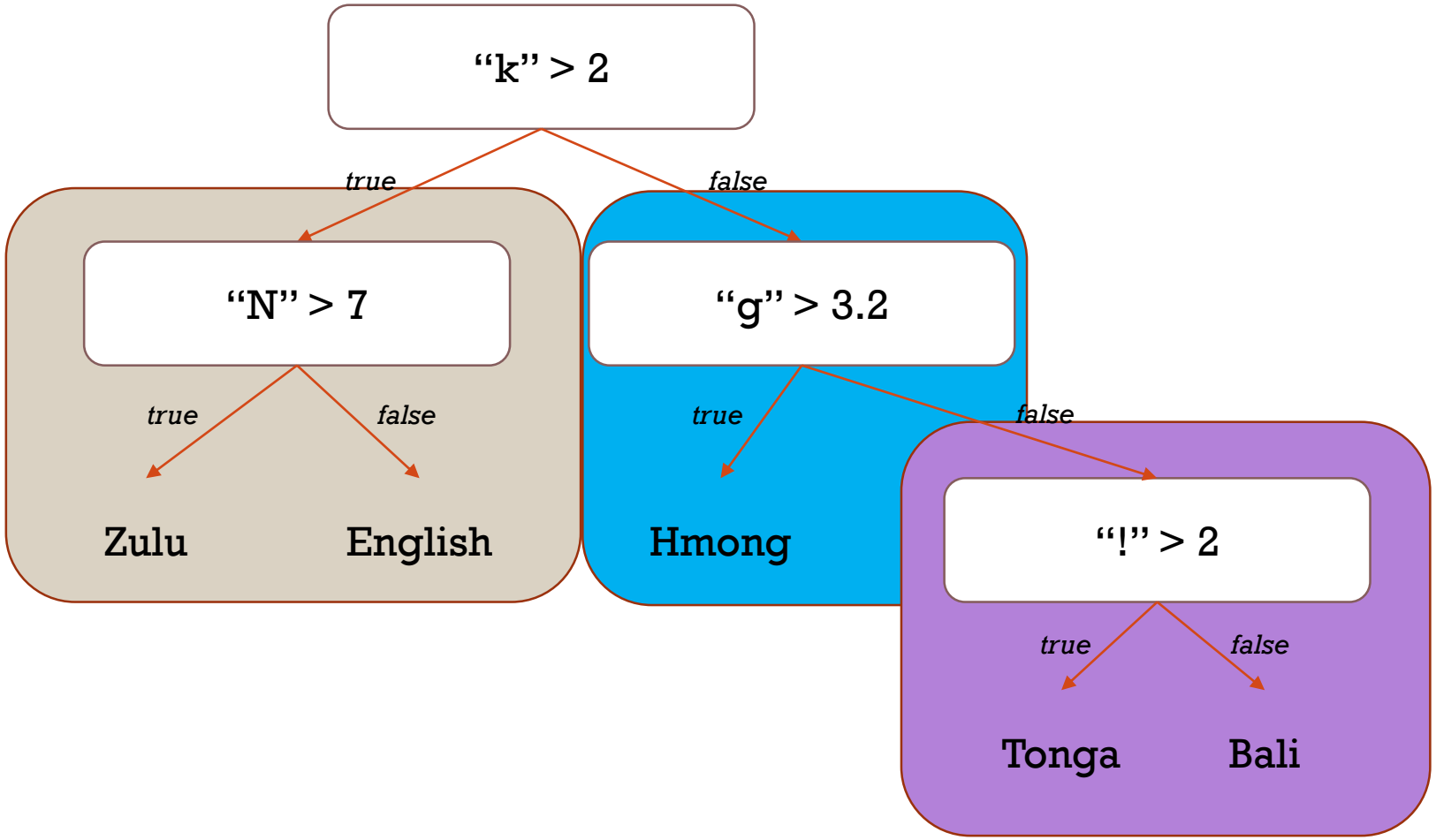
```
if count["g"] > 3.2:
    return "Hmong"
```

One way to predict is with a "*decision tree*"




```
if count["N"] > 7:
    return "Zulu"
else:
    return "English"


if count["g"] > 3.2:
    return "Hmong"
else:
```

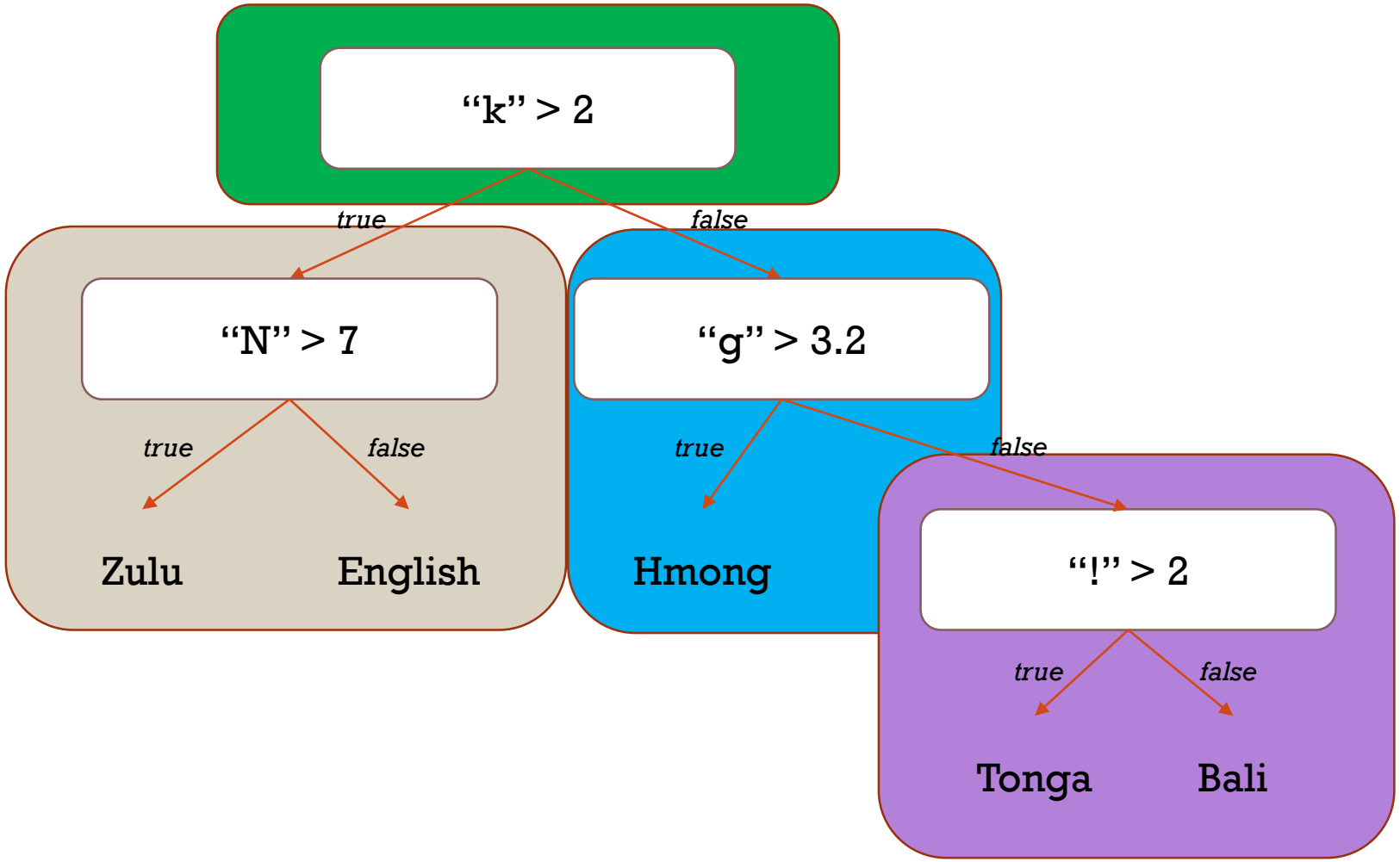
One way to predict is with a "*decision tree*"




```
if count["N"] > 7:
    return "Zulu"

else:
    return "English"
```

```
if count["g"] > 3.2:
    return "Hmong"

else:

    if count["!"] > 2:
        return "Tonga"

    else:
        return "Bali"
```

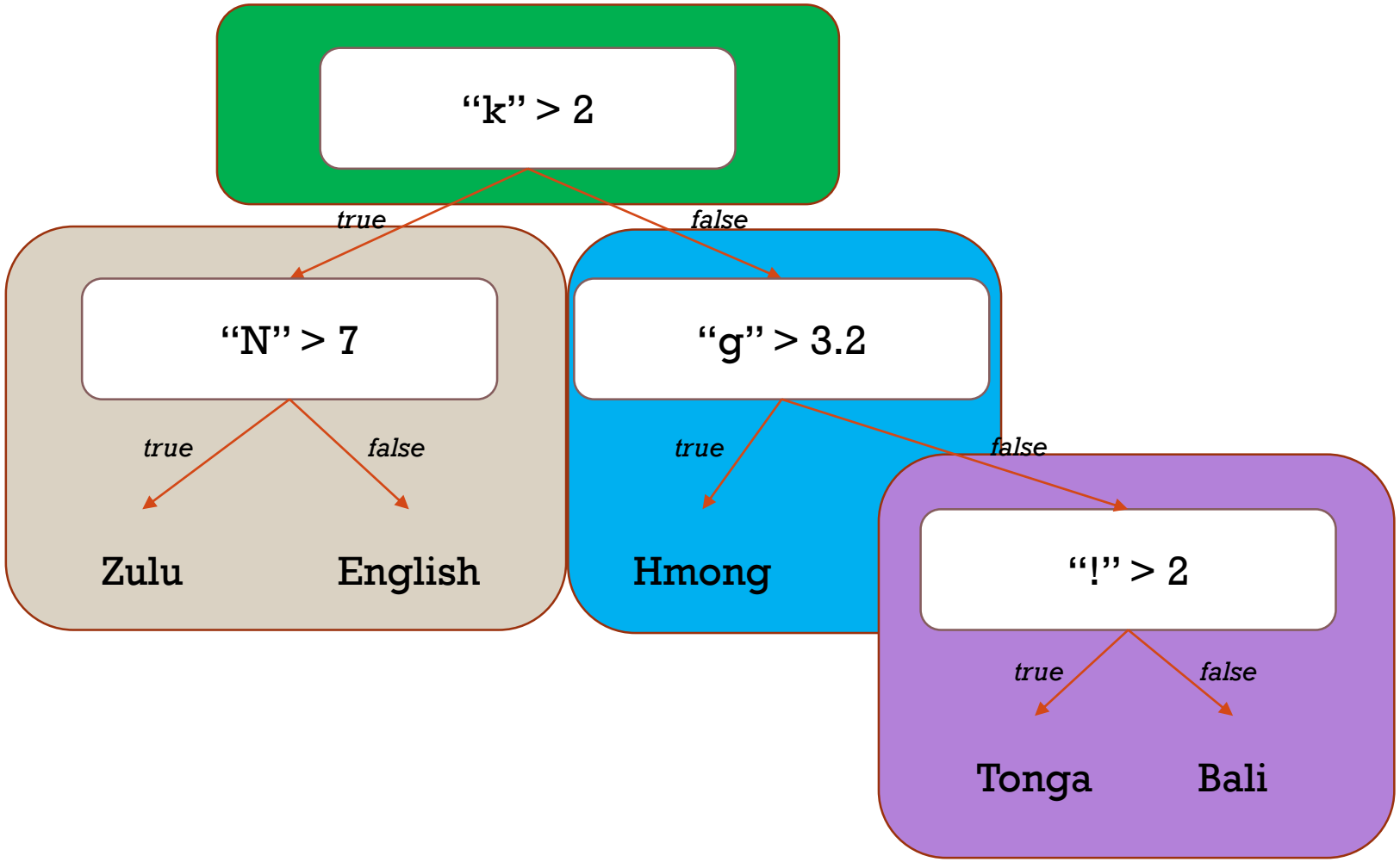
One way to predict is with a "*decision tree*"




```
if count["N"] > 7:
    return "Zulu"

else:
    return "English"
```

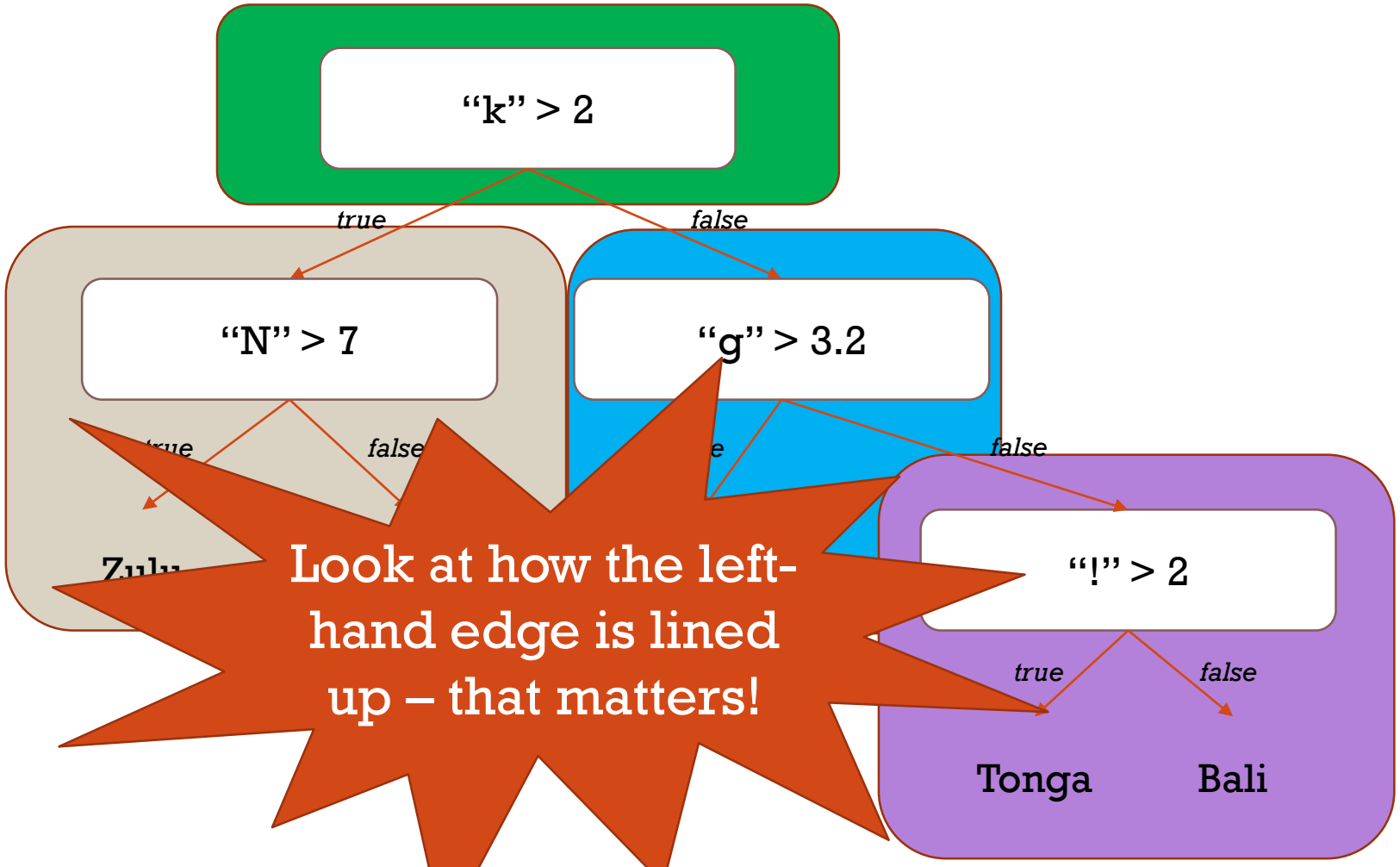
```
if count["g"] > 3.2:
    return "Hmong"

else:

    if count["!"] > 2:
        return "Tonga"

    else:
        return "Bali"
```

One way to predict is with a "*decision tree*"



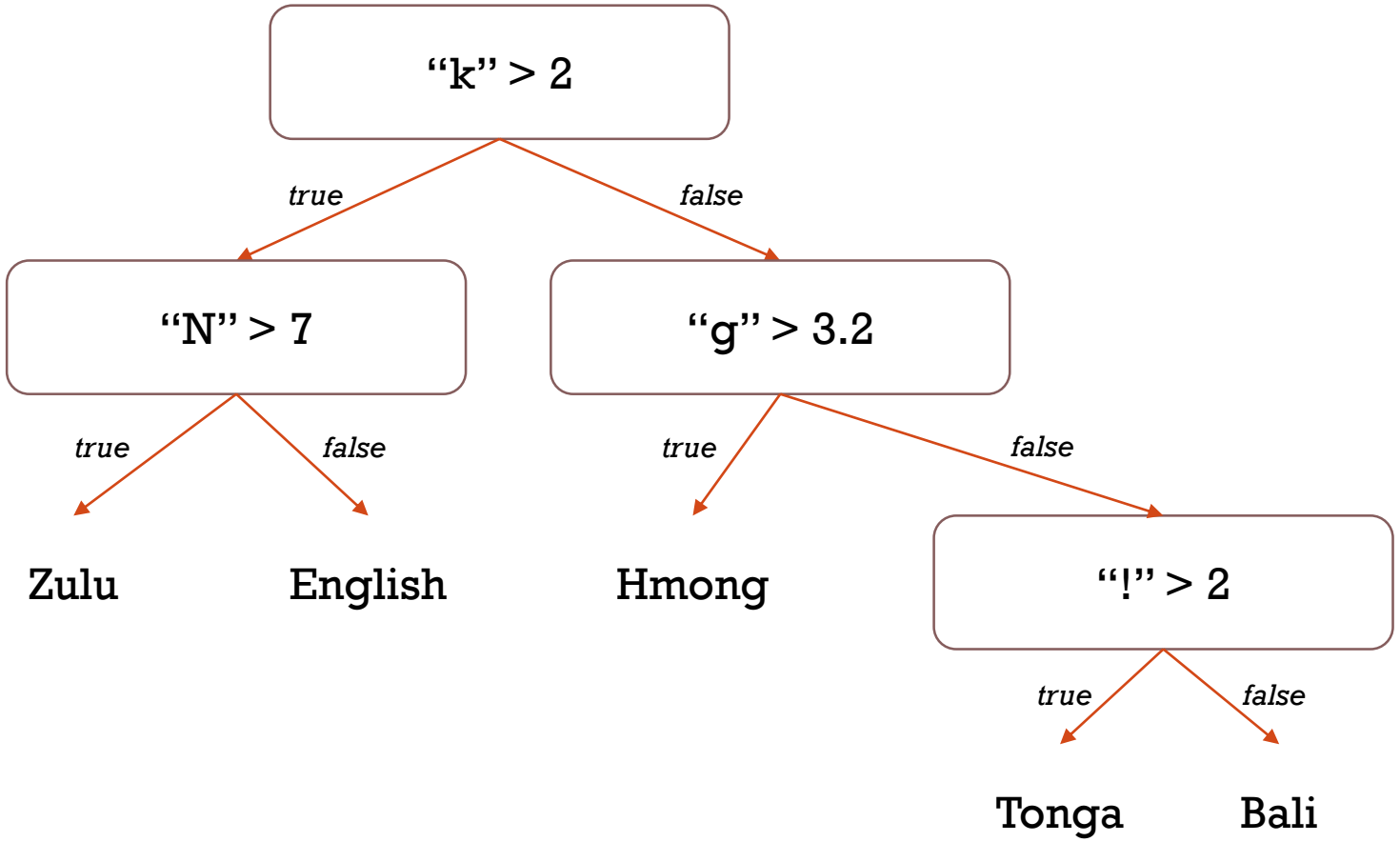

```
if count["k"] > 2:
    if count["N"] > 7:
        return "Zulu"
    else:
        return "English"
else:
    if count["g"] > 3.2:
        return "Hmong"
    else:
        if count["!"] > 2:
            return "Tonga"
        else:
            return "Bali"
```

One way to predict is with a "*decision tree*"




```
if count["k"] > 2:
    if count["N"] > 7:
        return "Zulu"
    else:
        return "English"
else:
    if count["g"] > 3.2:
        return "Hmong"
    else:
        if count["!"] > 2:
            return "Tonga"
        else:
            return "Bali"
```

One way to predict is with a "*decision tree*"




```
if count["k"] > 2:
    if count["N"] > 7:
        return "Zulu"
    else:
        return "English"
else:
    if count["g"] > 3.2:
        return "Hmong"
    else:
        if count["!"] > 2:
            return "Tonga"
        else:
            return "Bali"
```

We can build an *"algorithm"* (procedure/program) that plays the game of language identification

Train (Learning Phase):

- We get a lot of example texts in each language
- We tell the computer how to think about the data

Test (Evaluation Phase):

- When we give it a new text, it thinks about the data the same way…
- Then it predicts the most likely label
- We grade how good the tree is at predicting

Let's write a procedure (computer program) to identify a language…

- You have some example texts labeled with their languages
- Now someone hands you a new text without a label
- What do you do?

Let's write a procedure (computer program) to identify a language…

- You have some example texts labeled with their languages (document)
- Now someone hands you a new text without a label
- What do you do?

Let's write a procedure (computer program) to identify a language…

- You have some example texts labeled with their languages (document)
- Now someone hands you a new text without a label (`counts`)
- What do you do?

Let's write a procedure (computer program) to identify a language…

- You have some example texts labeled with their languages (document)
- Now someone hands you a new text without a label (`counts`)
- What do you do? (your Python code!)

Let's write a procedure (computer program) to identify a language…

- You have some example texts labeled with their languages (document)
- Now someone hands you a new text without a label (`counts`)
- What do you do? (your Python code)

Instructions

- ***Wait*** for some Python gotchas
- Then say hello to your neighbor
- *Build a tree:* Look at the document & choose (a) which letters to use and (b) how to nest them
- *Evaluate:* See how well the decision tree performs
- *Iterate:* Keep changing your tree and evaluating!

Be careful with Python syntax!

- Gotchas that you should double check…
  - Did you use a **colon** after every *if* and every *else*?
  - To get **count["a"]**, are you using square brackets, quotation marks, a single character, and the word *count* (no -s)?
  - Are your **tabs** where you want them?
  - Did you use the word **return** to make a prediction?
  - Did you **both open and close** the quotes, brackets, etc.?
  - Are there any **typos** in names? (including capitalization)
  - A **KeyError** means that that character isn't used. Try another one.

```
if count["k"] > 2:
    if count["N"] > 7:
        return "Zulu"
    else:
        return "English"
else:
    if count["g"] > 3.2:
        return "Hmong"
    else:
        if count["!"] > 2:
            return "Tonga"
        else:
            return "Bali"
```

Go forth!

Code:
https://repl.it/EW7e/1

Document with (language, text) pair data:
http://bit.ly/2etykDS

When we use decision trees for real…

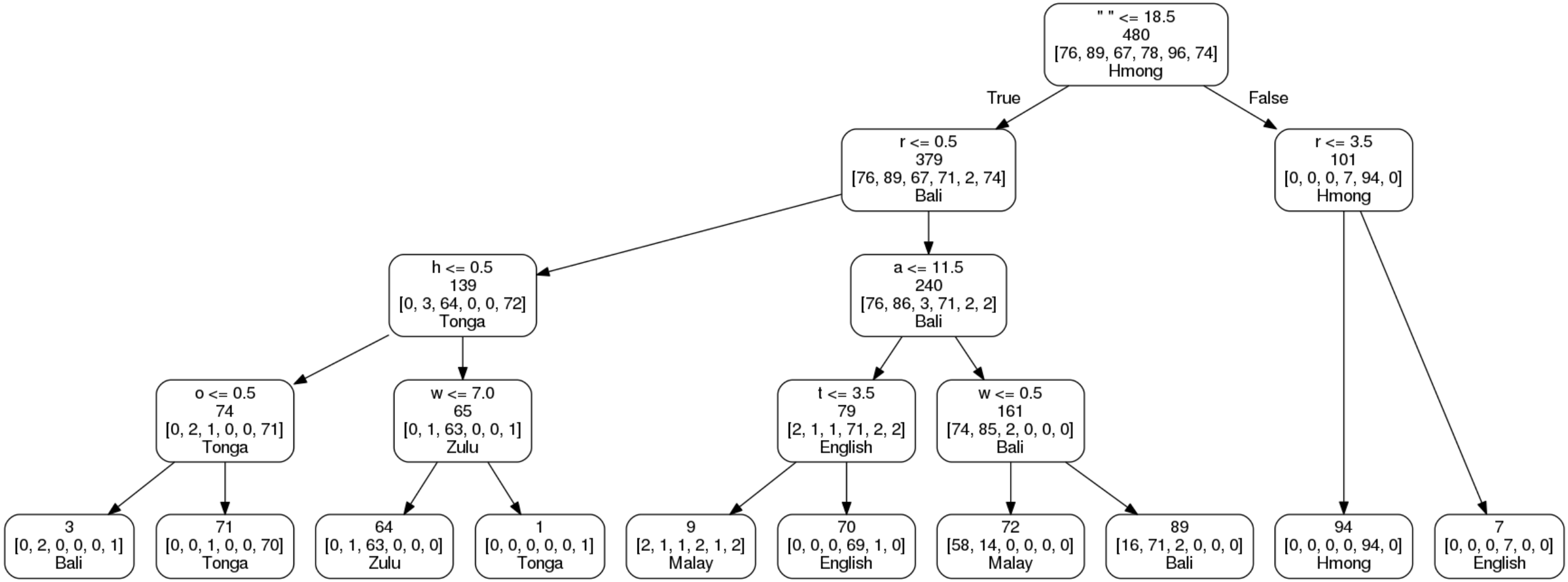
- We give the *computer* a lot of examples, and we ask the computer to figure out how to split the information

- Good properties of splits:
  - About half the data in each split
  - The groups after splitting are very uniform

- Computers can do this better than people… (why?)

- + There are a *lot* more methods than decision trees out there!

The computer gets 89.5% correct!



```
from collections import defaultdict
from sklearn import tree
from sklearn.feature_extraction import DictVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

def get_unigrams(string):
    unigrams = defaultdict(int)
    for character in string:
        unigrams[character] += 1
    return unigrams

# Create the dataset
languages = [("Malay", malay), ("Bali", bali), ("Zulu", zulu), ("English", english), ("Hmong", hmong), ("Tonga",
tonga)]
NUM_CHARS_IN_EXAMPLE = 100
list_of_dictionaries = []
y =[]
for langid, (lang, data) in enumerate(languages):
    for i in range(0,len(data),NUM_CHARS_IN_EXAMPLE):
        line_of_data = data[i:i+NUM_CHARS_IN_EXAMPLE]
        list_of_dictionaries.append(get_unigrams(line_of_data))
        y.append(langid)
X = DictVectorizer(sparse=False).fit_transform(list_of_dictionaries)

# Separate training from testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

# Build a classifier on the training data
clf = tree.DecisionTreeClassifier(max_depth=4)
clf = clf.fit(X_train, y_train)

# Evaluate the classifier on the test data
preds = clf.predict(X_test)
print accuracy_score(y_test, preds)
```

One step beyond:

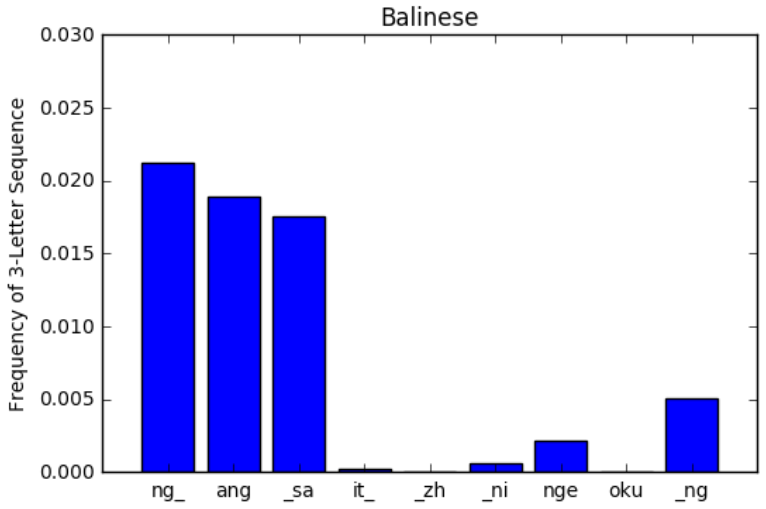
How can we score even better with exactly the same inputs?

(using a more complicated version of unigrams!)

The frequencies of letter sequences help

Balinese

Ring sajeroning pangawedar puniki tan dados katarka tur nguwehin inggian negara, paguyuban, utawi silih tunggal janma, …



Counts of letter sequences

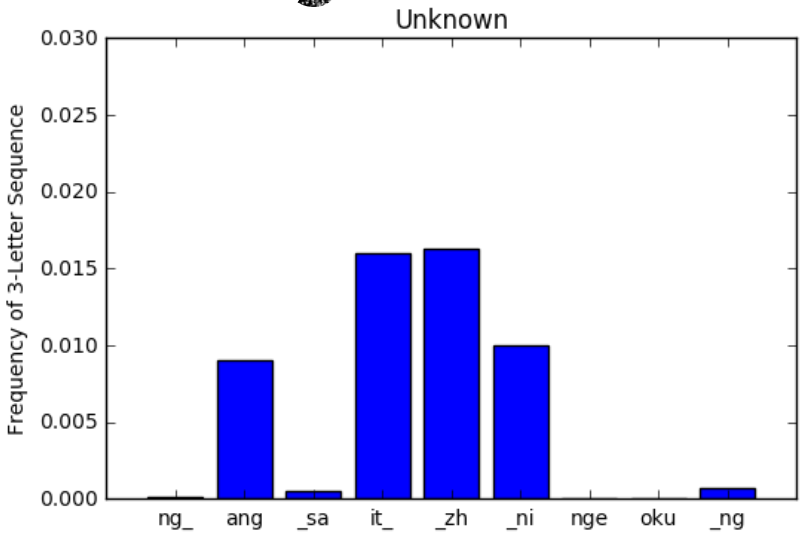
rin	1
ing	3
ng_	2
g_s	1
_sa	1

…

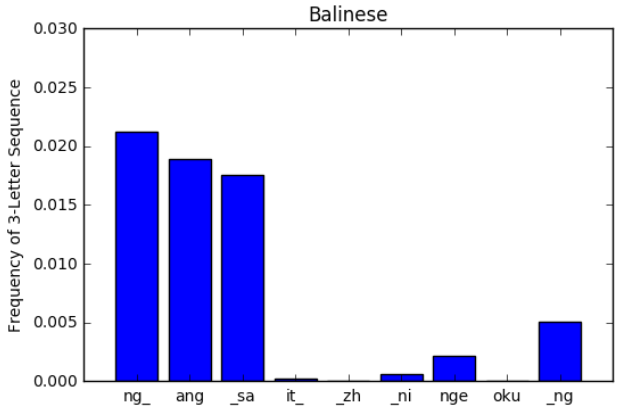
We call these 3-letter sequences "*trigrams*"

We can describe each language by its trigram counts -- in a histogram!

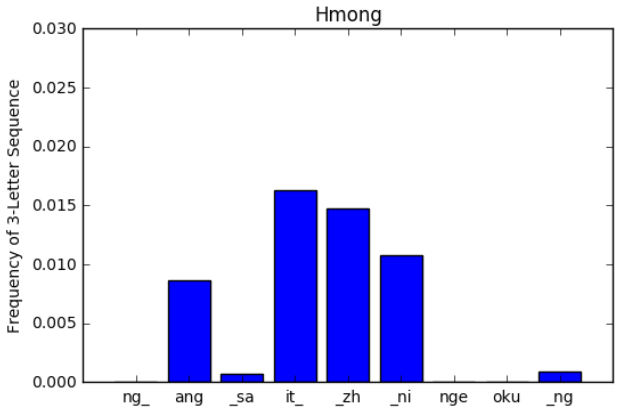
How hard is it to guess this language?



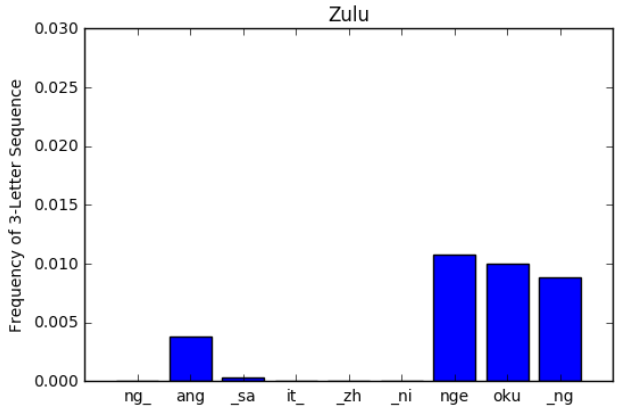
Balinese



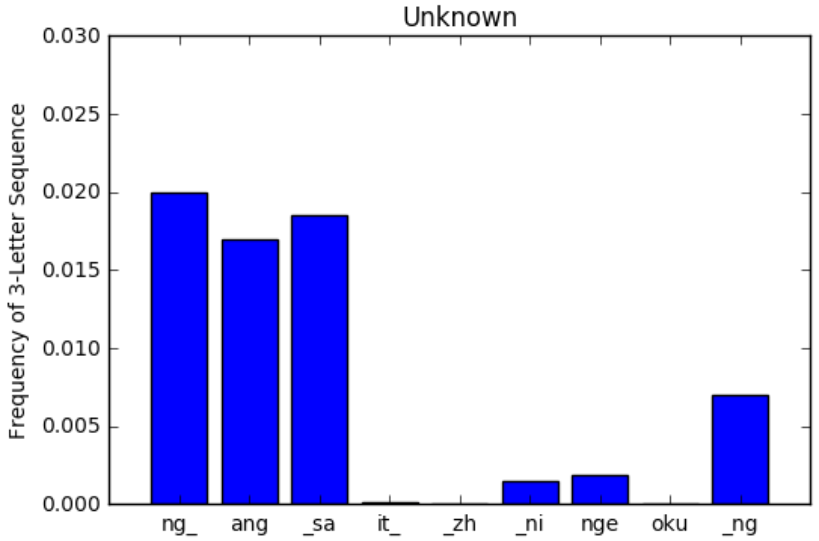
Hmong



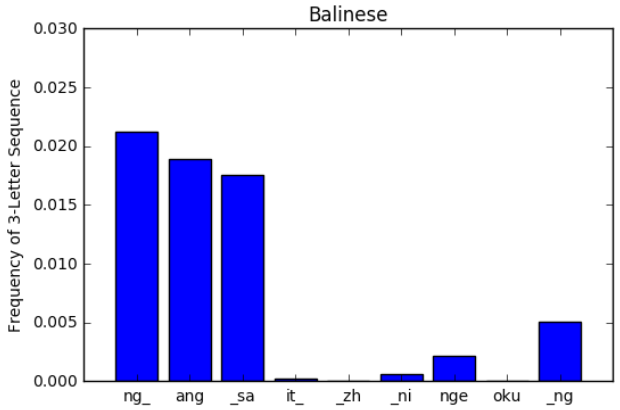
Zulu



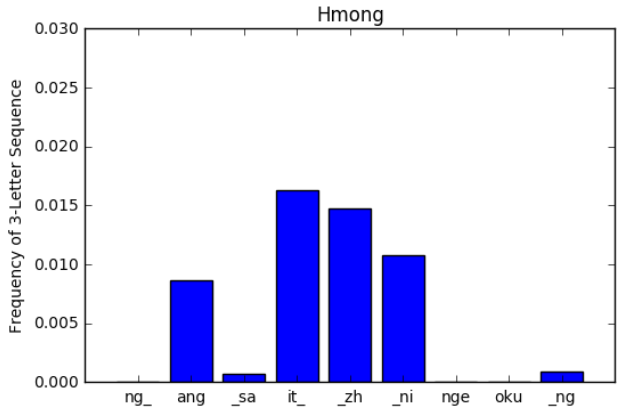
What about this one?



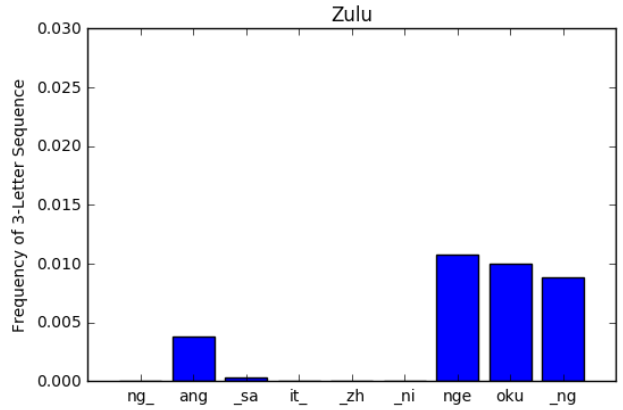
Balinese



Hmong



Zulu



Being smarter about inputs can pay off

- Sometimes it's possible to "cheat" and transform the input data….
- That's fair game!
- It often works pretty darn well
- And it's so common & important we have a specific term for it – "feature extraction"

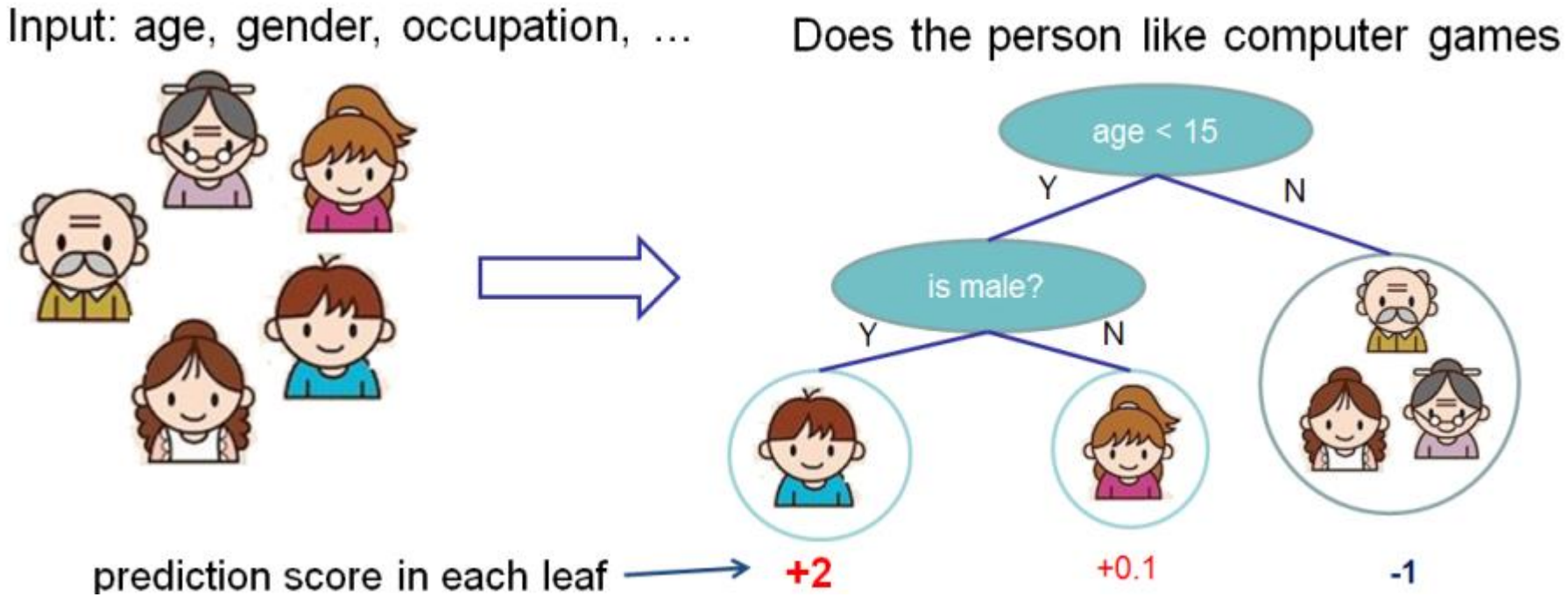
- The computer is phenomenal at repeating a boring task a zillion times – and humans are phenomenal at intuition.
- We do best if we use the strengths of each!

Real life story:

Who will leave our insurance company?

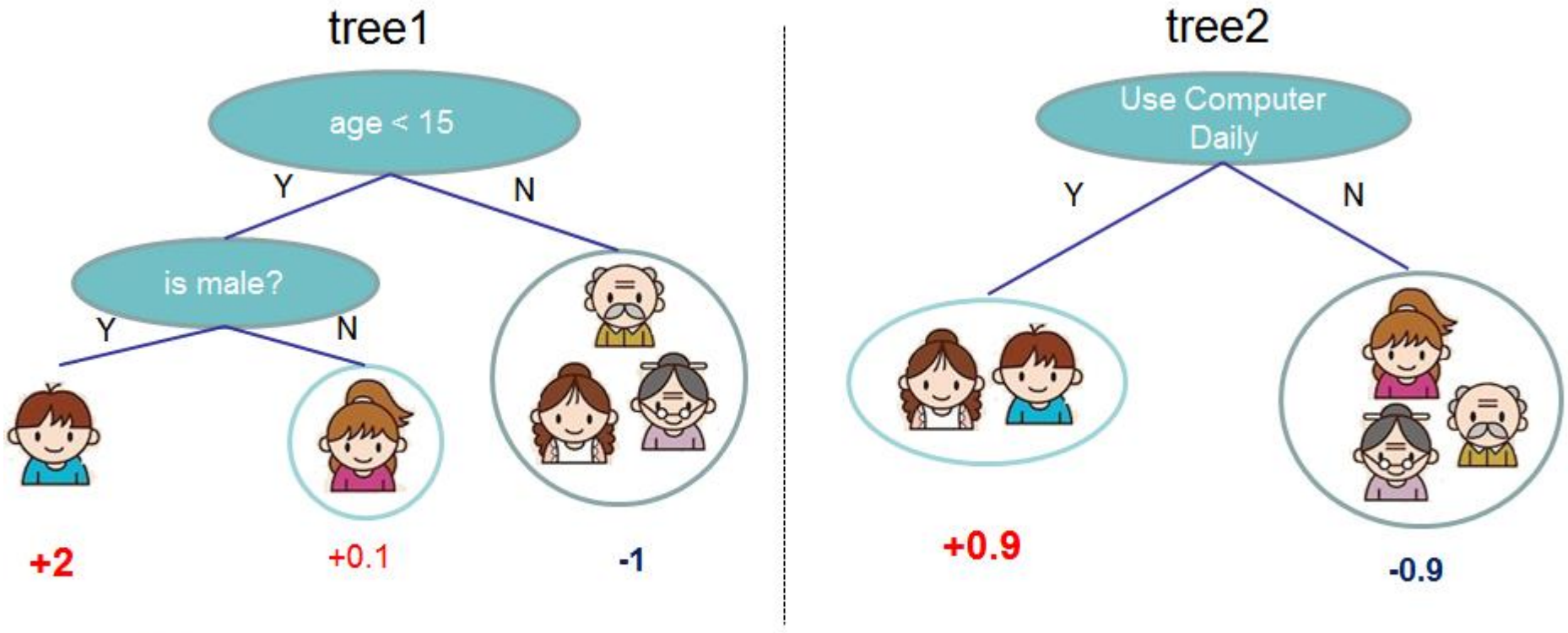
(using a more complicated version of trees!)

Another way to build a tree is put a number at the bottom & use a threshold



Positive number? We guess "you like games". *Negative number?* We guess "you don't".

We can combine multiple trees to get a better prediction



f( ) = 2 + 0.9= 2.9 f( )= -1 - 0.9 = -1.9

If we build new trees based on what we get *wrong*, we can do guess even better

- Since we have a limited amount of time to learn…
  - We want to ignore examples that are really easy to get right
  - We should spend our time getting better at the examples we still get wrong
  - If so, each tree we add will tend to improve the consensus opinion
- We've just described a "GBM" (Gradient Boosting Machine / Gradient Boosting Model) – it's one of the most powerful predictive techniques available today
- (Making guesses and learning from the results, with the aim of getting "less bad" all the time? That's a core idea in machine learning & AI. Maybe it's *the* core idea.)

What's next?

Going forward & deeper

- Stanford Splash (twice each year; next is December 3-4, 2016)
  - Grade 8: cryptography, computer graphics, graph theory, computer science with Java
  - Grade 7: graph theory, computer science with Java, logic, statistics
- Girls Who Code
- Computer History Museum
- code.org
- Undoubtedly more! Share ideas with each other.

- Pre-college AI is harder – most programs assume you know how to code, you enjoy math, and you have taken at least high school statistics.
  (but there is SAILORS & friends for rising 10th graders)