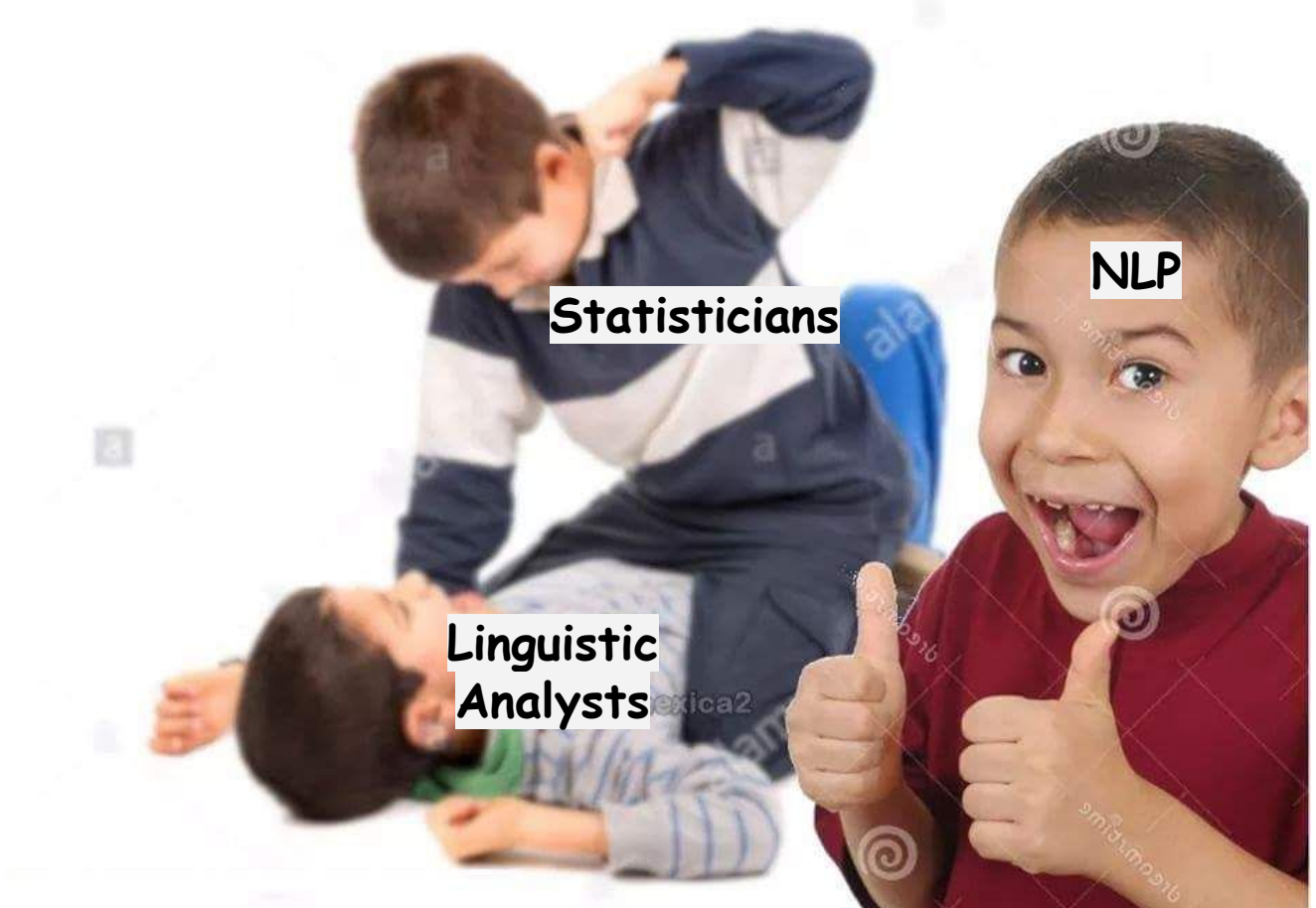
IDIOM DETECTION

GROUP F (06)

TABLE OF CONTENTS

HISTORY OF NLP

HISTORY OF NLP

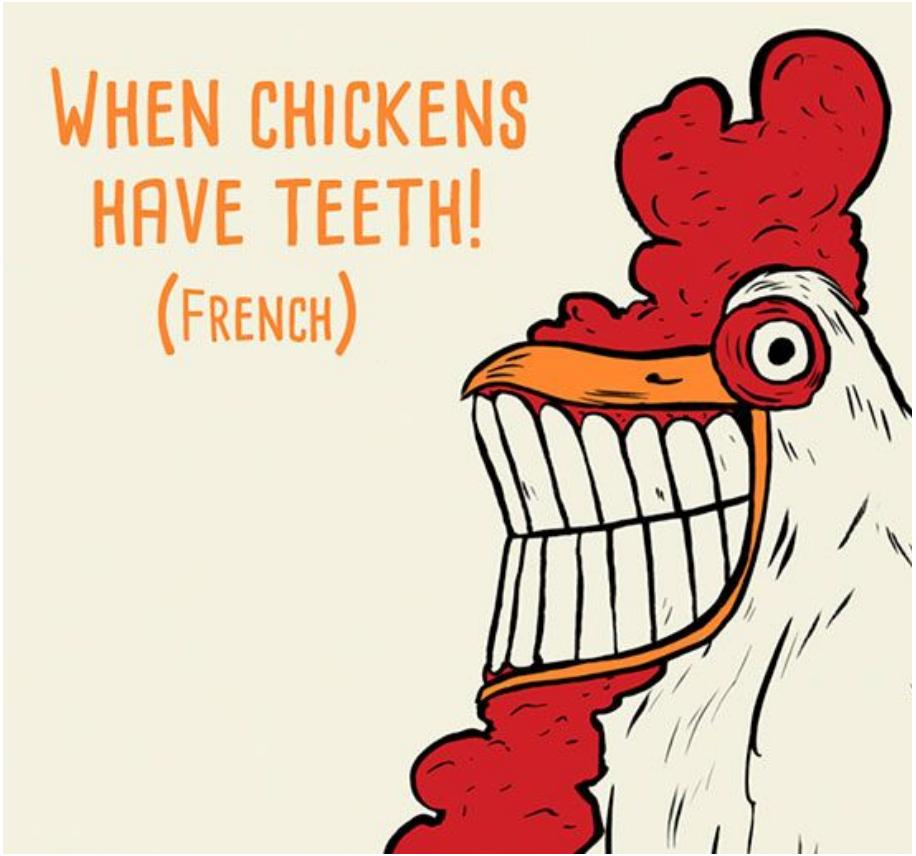


THE PROBLEM - WHAT IS AN IDIOM

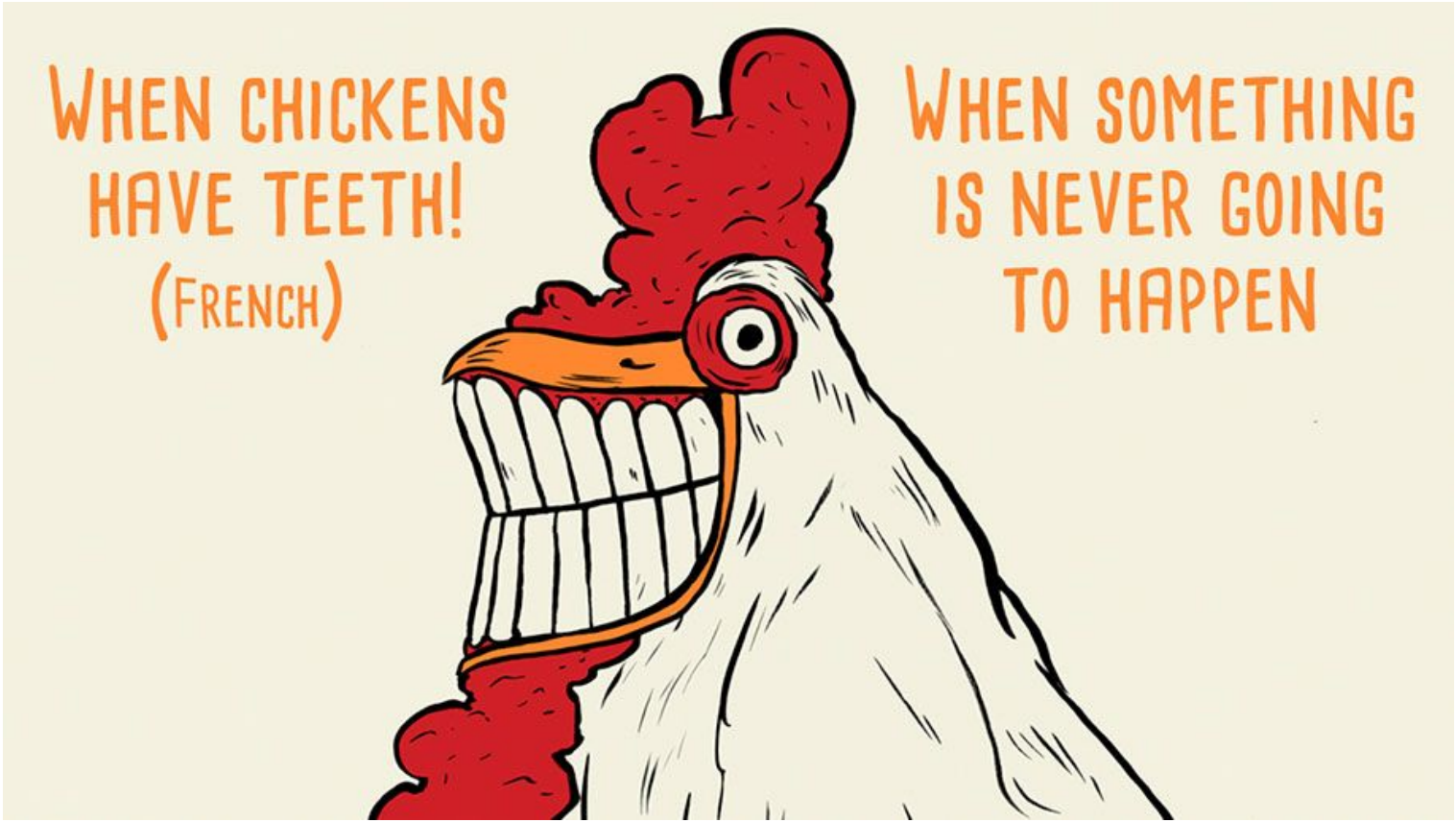
An idiom is a:

MULTIWORD EXPRESSION (MWE) **+** **WHOSE MEANING IS NON-COMPOSITIONAL**

THE PROBLEM



THE PROBLEM


WHEN CHICKENS HAVE TEETH! (FRENCH)
WHEN SOMETHING IS NEVER GOING TO HAPPEN

GRAND VISION

Technology

Globalisation

"The new era of global flows"
McKinsey Global Institute - 2016

News

85%

of adults report regularly reading news on mobile

Society

The End of Ownership

Language

0.7%

of students feel fluent in the language taught in high school and college

Flipboard

72 Million Active Users

feedly

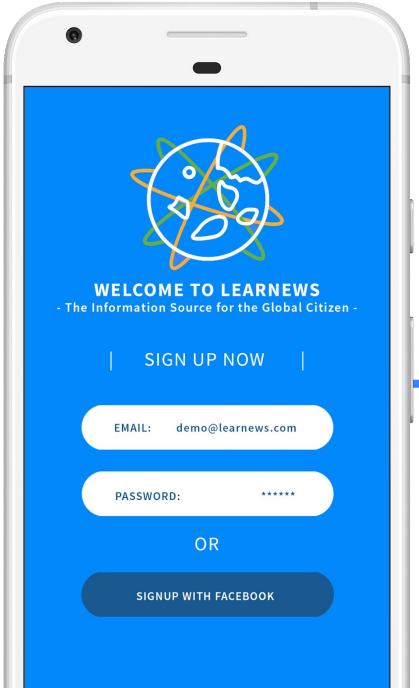
8 Million Users

duolingo

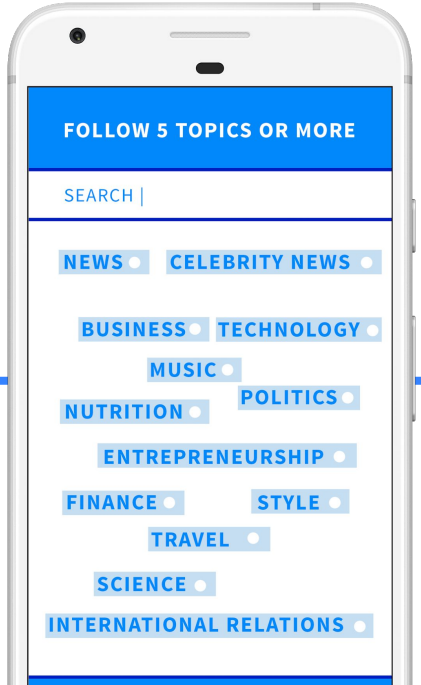
+100 Million Users

GRAND VISION

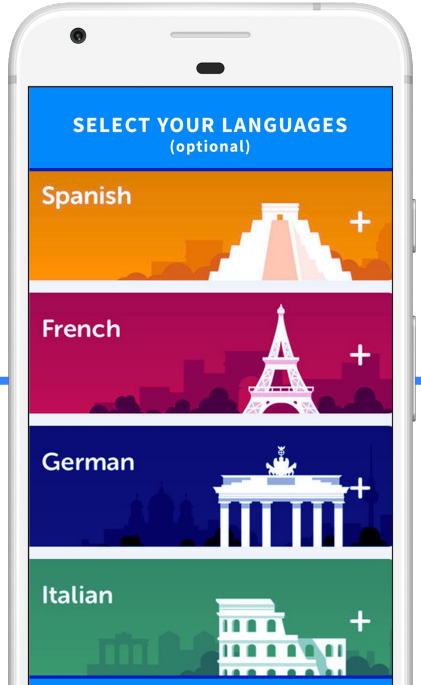
Join a Community of Global Citizens



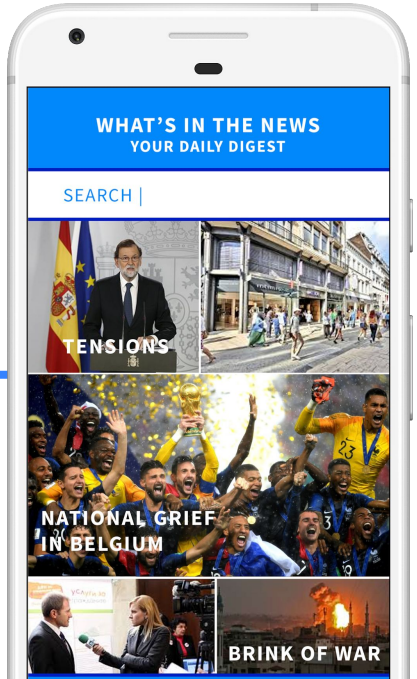
Select the Topics in your Feed



Select the Languages in your Feed



Discover Premium Quality Articles



GRAND VISION

June 2019

NLP Research

The most challenging part, being able to dissociate an idiom from a random string of words is essential to provide a pleasant reading experience

August 2019

Translation & Definition Research

Connecting the identified words and idioms to a translation or definition to allow building the MVP

September 2019

Partnership Development

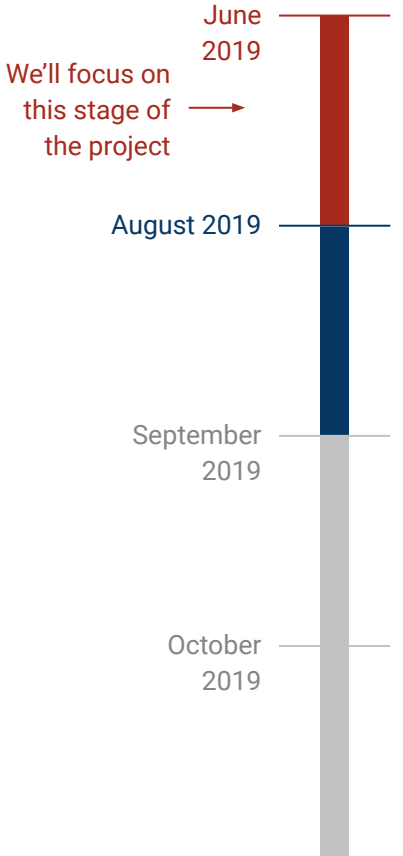
Developing partnerships with newspapers offering them the opportunity to broaden their audience with international readers

October 2019

App Development

Working on getting the app ready to launch for Q1 2020

GRAND VISION



NLP Research

The most challenging part, being able to dissociate an idiom from a random string of words is essential to provide a pleasant reading experience

Translation & Definition Research

Connecting the identified words and idioms to a translation or definition to allow building the MVP

Partnership Development

Developing partnerships with newspapers offering them the opportunity to broaden their audience with international readers

App Development

Working on getting the app ready to launch for Q1 2020

GRAND VISION

Young and Free: Prerequisite for Success

Let's face it: In today's business world you need to be young and free of attachments to strike it rich. It's a dog eat dog world out there and you're going to have to work quite a lot. Of course, not only will you have to work quite a lot, you'll need to be flexible and ready to take advantage of anything. That's where the "free" part comes in.

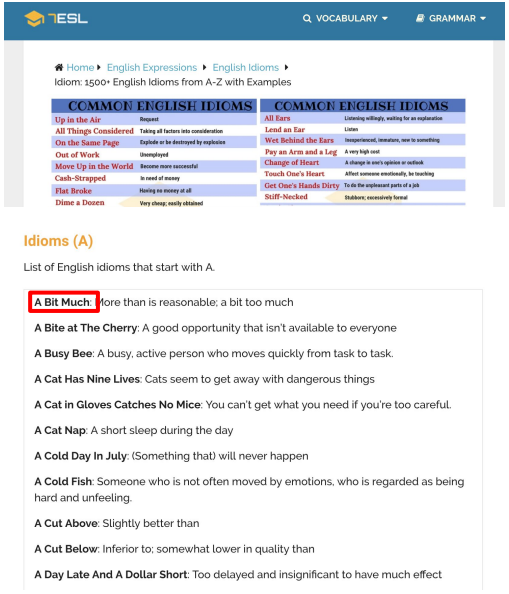
I've got a young friend, he's only 25, but he fits the bill perfectly. He's single and he's hungry. He's willing to start from scratch and, best of all, he isn't afraid of putting his nose to the grindstone for those 80 hour weeks. He decided to take the bull by the horns by going starting up his own business. He found a software developer who knew the internet inside out. This young man was also very ambitious. He left his safe job at the drop of a hat. They were both reaching for pie in the sky, and they were ready.

Identify Highlight → Describe or Translate


OK FOLKS
LET'S DIVE IN.
memegenerator.net

WEB SCRAPING - IDIOMS

SOURCE



SCRAPER

```python
# -*- coding: utf-8 -*-

import pandas as pd
import requests
import re
import json
import unicodedata
# from utils import clear_file
from bs4 import BeautifulSoup

OUTPUT_FILE = "idioms.txt"

response = requests.get("https://7esl.com/english-idioms/")
soup = BeautifulSoup(response.text, "lxml")
r_text = soup.find_all('li')
# .get_text(strip=True)

# r_text = response.text.encode('utf-8')

idioms = []
sentences = []
for l in r_text:
  if re.search('<li><em>',str(l)):
#     print(l)
    keyword = re.findall('<strong>(.*?)<\/strong>',str(l))
    if len(keyword):
      keyword = re.sub('<em>|<\/em>|\(','',keyword[0])
    else:
      keyword = re.findall('<strong style=\"font-style: inherit;\">(.*?)</strong>', str(l))
      keyword = re.sub('<em>|<\/em>|\(','',keyword[0])

    sentence = re.sub('(<\w+>)|(<\/\w+>)|(\(.*?\))|(<strong style=\"font-style: inherit;\">)','',str(l))
    sentence = re.sub('(\s+)',' ',sentence)
    idioms.append(keyword)
    sentences.append(sentence)
    print(keyword)
    print(sentence)
    print("==============================")

example_df = pd.DataFrame({'idiom':idioms, 'sentence':sentences})
example_df['idiom'] = example_df['idiom'].map(lambda x: re.sub('\(','',str(x)))
example_df.to_csv('/data/idiom_example.csv')
```

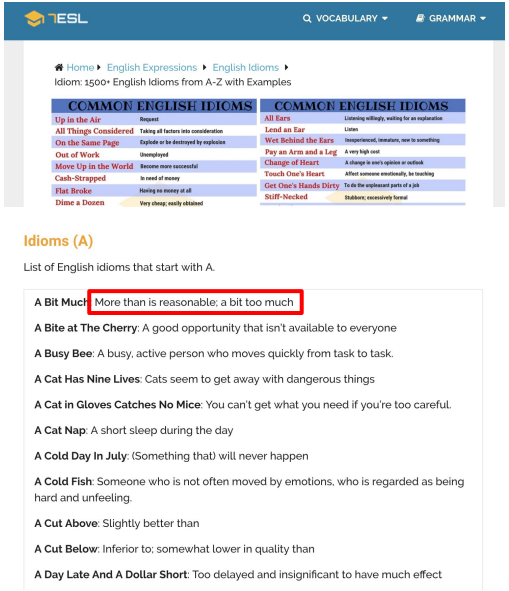
OUTPUT

1546 lines (1545 sloc) | 29 KB

```
A Bit Much
A Bite at The Cherry
A Busy Bee
A Cat Has Nine Lives
A Cat in Gloves Catches No Mice
A Cat Nap
A Cold Day In July
A Cold Fish
A Cut Above
A Cut Below
A Day Late And A Dollar Short
A Dog in The Manger
A Few Sandwiches Short Of A Picnic
A Good Deal
A Great Dea
A Guinea Pig
A Hair's Breadth
A Home Bird
A Hundred And Ten Percent
A Lame Duck
A Leg Up
A Lemon
A Life Of Its Own
A Little Bird Told Me
A Little Bird Told Me
A Little from Column A, a Little from Column B
A Lone Wolf
A Lot on One's Plate
A Million and One
```

WEB SCRAPING - SENTENCES WITH IDIOMS

SOURCE



SCRAPER

```python
# -*- coding: utf-8 -*-

import pandas as pd
import requests
import re
import json
import unicodedata
# from utils import clear_file
from bs4 import BeautifulSoup

OUTPUT_FILE = "idioms.txt"

response = requests.get("https://7esl.com/english-idioms/")
soup = BeautifulSoup(response.text, "lxml")
r_text = soup.find_all('li')
# .get_text(strip=True)

# r_text = response.text.encode('utf-8')

idioms = []
sentences = []
for l in r_text:
  if re.search('<li><em>',str(l)):
#     print(l)
    keyword = re.findall('<strong>(.*?)</strong>',str(l))
    if len(keyword):
      keyword = re.sub('<em>|</em>|\(','',keyword[0])
    else:
      keyword = re.findall('<strong style=\"font-style: inherit;\">(.*?)</strong>', str(l))
      keyword = re.sub('<em>|</em>|\(','',keyword[0])

    sentence = re.sub('(<\w+>)|(<\/\w+>)|(\(.*?\))|(<strong style=\"font-style: inherit;\">)','',str(l))
    sentence = re.sub('(\s+)',' ',sentence)
    idioms.append(keyword)
    sentences.append(sentence)
    print(keyword)
    print(sentence)
    print("================================")

example_df = pd.DataFrame({'idiom':idioms, 'sentence':sentences})
example_df['idiom'] = example_df['idiom'].map(lambda x: re.sub('\(','',str(x)))
example_df.to_csv('/data/idiom_example.csv')
```

OUTPUT

240 lines (239 sloc) | 22.9 KB

Search this file...

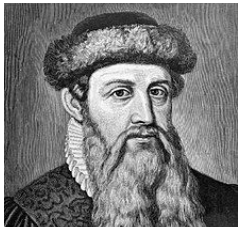
	idiom	sentence
0	as pale as a ghost	My grandfather was as pale as a ghost when
1	at death's door	The sales manager was at death's door after
2	back on her feet	My mother is back on her feet after being sic
3	feeling on top of the world	I have been feeling on top of the world since
4	going under the knife	I'm going under the knife next month to try to
5	green around the gills	My colleague was looking a little green aroun
6	has one foot in the grave	My uncle is very sick and has one foot in the
7	sick as a dog	Did you have a good vacation? – Not really. I
8	under the weather	My boss has been under the weather all week
9	all the rage	A few years ago Uggs were all the rage , but
10	at the drop of a hat	Jacob is unpredictable. He won't leave the of
11	knock your socks off	Wait until you try the new Yamaha scooters.
12	old hat	The carmaker's sales declined because many
13	ballpark figure	A ballpark figure for the cost of the new stadi
14	hit it out of the park	Francesca hit it out of the park with her spee
15	kicked ass	Madrid won most of our matches during the s
16	on deck	I'll call you back in an hour. The speaker is al
17	second wind	I thought I was totally exhausted after mile ni

CORPORA USED - 210k sentences

BROWN

The Brown Corpus was the **first million-word electronic corpus of English**, created in 1961 at Brown University.

57,340 sentences



A small selection of texts from the Project Gutenberg archive, which contains some **25,000 free electronic books**.

98,552 sentences

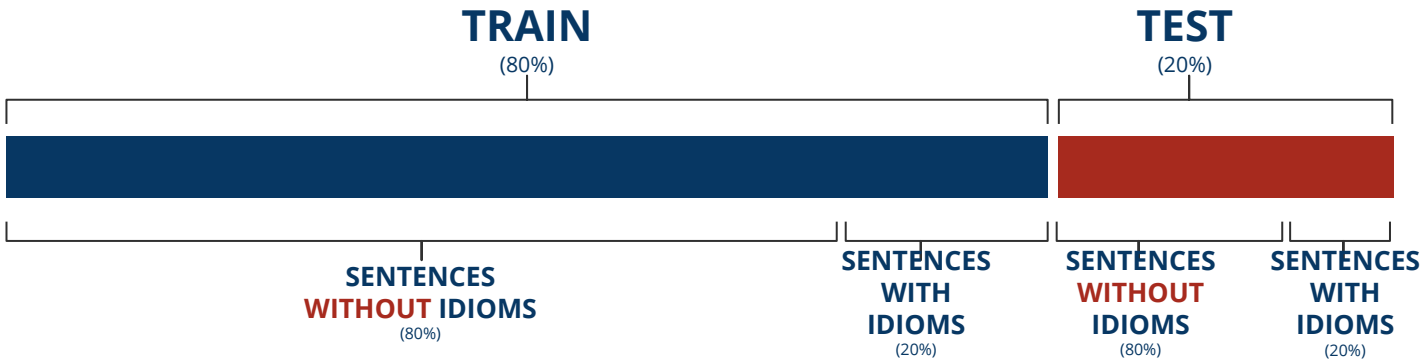
REUTERS®

The Reuters Corpus **contains 10,788 news documents** (1.3 million words). The documents are classified into 90 topics.

54,716 sentences

WEB SCRAPING - PREPARING THE DATA

```
[('You', 'PRP', 'OUT'),
 ("don't", 'VBP', 'OUT'),
 ('want', 'VB', 'OUT'),
 ('to', 'TO', 'OUT'),
 ('lease', 'VB', 'OUT'),
 ('a', 'DT', 'OUT'),
 ('place', 'NN', 'OUT'),
 ('way', 'NN', 'OUT'),
 ('out', 'IN', 'BEGIN'),
 ('in', 'IN', 'IN'),
 ('the', 'DT', 'IN'),
 ('sticks', 'NNS', 'IN'),
 ('.', '.', 'OUT')]
```



NLP

Words surrounding an idiomatic expression are less similar when used in a figurative sense

NLP

```
{'bias': 1.0,
 'word.lower()': 'anybody',
 'word[-3:]': 'ody',
 'word.isupper()': False,
 'word.istitle()': True,
 'word.isdigit()': False,
 'postag': 'NN',
 'postag[:2]': 'NN',
 'BOS': True,
 '+1:word.lower()': 'who',
 '+1:word.istitle()': False,
 '+1:word.isupper()': False,
 '+1:postag': 'WP',
 '+1:postag[:2]': 'WP',
 '+1:word2vec': 0.311652,
 '+2:word.lower()': 'is',
 '+2:word.istitle()': False,
 '+2:word.isupper()': False,
 '+2:postag': 'VBZ',
 '+2:postag[:2]': 'VB',
 '+2:word2vec': 0.2359705,
 '+3:word.lower()': 'expecting',
 '+3:word.istitle()': False,
 '+3:word.isupper()': False,
 '+3:postag': 'VBG',
 '+3:postag[:2]': 'VB',
 '+3:word2vec': 0.22183287}
```

- Lemmatizing preprocessing for Word2Vec

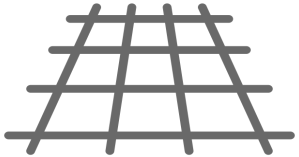
```
In [43]: WORD2VEC.similarity("funny","Antoine")
Out[43]: 0.05817873

In [44]: WORD2VEC.similarity("funny","Dan")
Out[44]: 0.13197193
```

- Unigram and Bigram Frequencies for PMI and PPMI

$$\mathrm{PPMI}(w,c) = \max\left(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0\right)$$

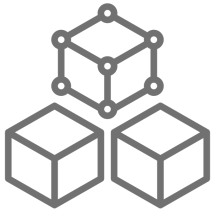
- Case formats: Title / Upper / Lower
- POS-tagging

NLP



Grid Search on:

- Number of words ahead/behind
- PPMI similarity score
- Word2Vec similarity score



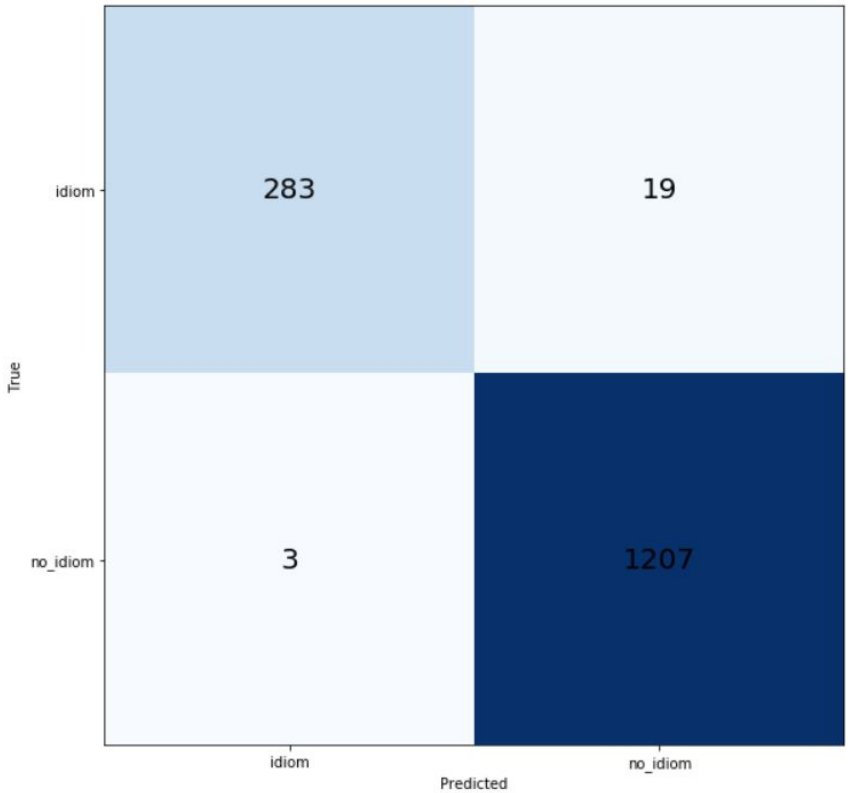
Model: Conditional Random Field (CRF)

- LBGFS Solver
- Maximum 100 iterations

MODELLING

From \ To	BEGIN	IN	IN's	OUT
BEGIN	0.0	6.894	0.0	-3.569
IN	0.0	5.963	0.393	-1.706
IN's	0.0	0.0	0.0	-1.321
OUT	2.145	0.0	0.0	4.0

	precision	recall	f1-score	support
BEGIN	0.990	0.937	0.963	302
IN	0.989	0.936	0.962	297
OUT	1.000	1.000	1.000	1512
micro avg	0.997	0.982	0.989	2111
macro avg	0.993	0.958	0.975	2111
weighted avg	0.997	0.982	0.989	2111
samples avg	0.999	0.992	0.993	2111



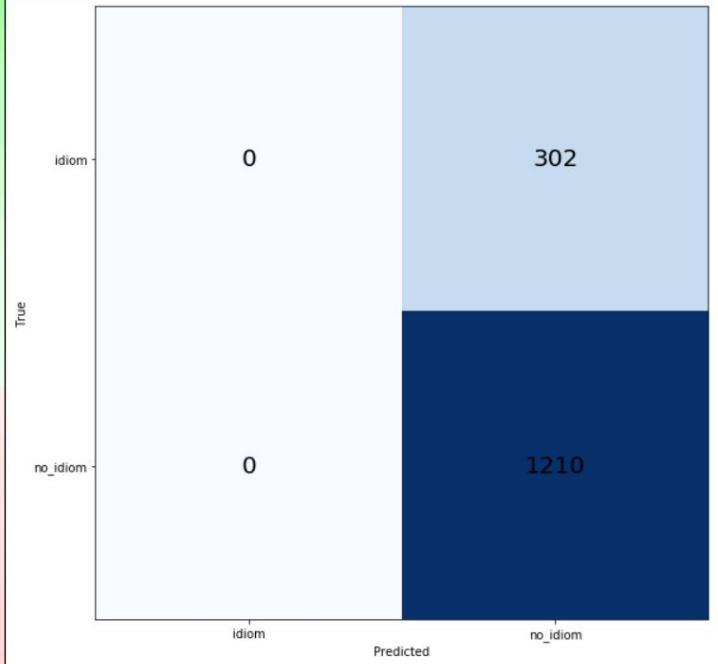
MODELLING

MODELLING

y=BEGIN top features		y=IN top features		y=IN's top features		y=OUT top features	
Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature
+6.638	word.lower():crickets	+2.535	-1:word.lower():poison	+0.937	-3:word.lower():end	+4.246	EOS
+2.900	+2:word.lower():deal	+2.535	word.lower():pill	+0.686	word.lower():day's	+2.412	-2:word.lower():about
+2.759	+1:word.lower():touch	+2.525	-1:word.lower():grips	+0.685	+1:word.lower():stint	+1.974	+3:word.lower():dark
+2.648	word[-3:]:ets	+2.299	word.lower():scratch	+0.660	word[-3:]:y's	+1.782	-1:word.lower():dark
+2.518	word.lower():tongue-in-cheek	+2.178	-1:word.lower():grease	+0.350	+2:word.lower():.	+1.736	BOS
+2.479	+1:word.lower():scratch	+2.178	word.lower():monkey	+0.344	+2:postag:.	+1.648	postag[:2]:.
+2.367	word.lower():about	+2.085	word.lower():counting	+0.344	+2:postag[:2]:.	+1.648	postag:.
+2.274	+1:word.lower():change	+1.878	word[-3:]:Red	+0.303	+1:ppmi	+1.632	-2:word.lower():good
+2.251	+2:word.lower():dark	+1.798	+2:word.lower():air	+0.255	-3:postag:NN	+1.631	-1:word.lower():along
+2.211	+1:word.lower():pill	+1.763	+3:word.lower():n't	+0.253	-2:word.lower():of	+1.594	postag::
+2.211	word.lower():poison	+1.756	-1:word.lower():blow	+0.247	-3:ppmi	+1.594	postag[:2]::
+2.174	word.lower():grease	+1.619	-1:word.lower():lose	+0.188	-3:postag[:2]:NN	+1.542	-3:word.lower():dlrs
+2.174	+1:word.lower():monkey	+1.616	-1:word.lower():eleventh	+0.174	-1:word.lower():the	*… 2560 more positive …*	
+2.161	word[-3:]:All	+1.594	-1:word.lower():head	+0.104	-2:postag:IN	*… 3050 more negative …*	
+2.093	+1:word.lower():counting	+1.556	word.lower():jazz	+0.104	-2:postag[:2]:IN	-1.554	word.lower():blow
+2.057	+3:word.lower():run	+1.556	word[-3:]:azz	+0.035	-1:postag[:2]:DT	-1.570	word[-3:]:oys
+2.040	+2:word.lower():nutshell	+1.551	+1:word.lower():forth	+0.035	-1:postag:DT	-1.585	-3:word.lower():n't
+1.920	+1:word.lower():elephant	+1.532	+1:word.lower():grapevine	+0.001	postag:NN	-1.590	-3:word.lower():'s
+1.904	+2:word.lower():spot	+1.530	-1:word.lower():about	+0.000	+1:postag:NN	-1.593	+3:word.lower():mouth
+1.866	+3:word.lower():air	+1.525	word.lower():lemon	-0.055	postag[:2]:NN	-1.609	-1:word.lower():lose
+1.831	+1:word.lower():axe	+1.519	+1:word.lower():spot	-0.077	+1:word.istitle()	-1.611	+1:word.lower():associates
+1.733	+2:word.lower():principle	+1.519	word.lower():shooting	-0.151	-2:ppmi	-1.646	word.lower():play
+1.683	+2:word.lower():forth	+1.510	word.lower():grapevine	-0.166	+1:postag[:2]:NN	-1.660	-1:word.lower():blow
+1.676	+1:word.lower():along	+1.502	word.lower():grips	-0.209	word.istitle()	-1.680	+3:word.lower():n't
+1.655	+3:word.lower():while	+1.501	+1:word.lower():nutshell	-0.852	+2:ppmi	-1.730	word[-3:]:ack
+1.647	+1:word.lower():lemon	+1.496	+1:word.lower():principle	-1.721	bias	-1.760	-2:word.lower():'re
+1.625	word[-3:]:eek	+1.488	word.lower():elephant			-1.828	-3:word.lower():seems
+1.622	word.lower():blow	*… 2625 more positive …*				-1.947	word.lower():change
… 1754 more positive …		*… 557 more negative …*				-2.063	word[-3:]:ets
… 383 more negative …		-1.526	-1:word.lower():that			-2.073	+3:word.lower():team
-1.754	+1:word.lower():on	-2.665	EOS			-2.476	word.lower():red
-1.976	+1:word.lower():that	-2.788	-2:word.lower():about			-2.634	word.lower():mother

MODELLING

y=BEGIN top features		y=IN top features		y=IN's top features		y=OUT top features	
Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature
+0.246	word[-3:]:out	+0.374	postag:NN	-0.098	+2:ppmi	+0.698	EOS
+0.214	word.lower():and	+0.289	-1:word.lower():all	-0.129	+1:ppmi	+0.594	BOS
+0.195	+2:postag:NN	+0.275	-1:word.lower():and	-0.699	bias	+0.475	-1:postag[:2]:NN
+0.173	-1:postag[:2]:VB	+0.258	-1:postag[:2]:IN			+0.382	bias
+0.142	postag[:2]:CC	+0.258	-1:postag:IN			+0.350	word.istitle()
+0.142	postag:CC	+0.245	-1:postag:DT			+0.333	postag:.
+0.126	+1:word.lower():all	+0.245	-1:postag[:2]:DT			+0.333	postag[:2]:.
+0.126	postag[:2]:IN	+0.229	+1:postag:NN			+0.237	word[-3:]:.
+0.126	postag:IN	+0.185	-1:postag:CC			+0.237	word.lower():.
+0.121	+1:ppmi	+0.185	-1:postag[:2]:CC			+0.177	postag[:2]:VB
+0.112	word[-3:]:and	+0.150	postag[:2]:NN			+0.097	postag[:2]:PR
+0.100	+2:word.lower():that	+0.137	-2:word.lower():and			+0.080	+1:ppmi
+0.092	+2:ppmi	+0.128	-3:postag[:2]:VB			+0.075	-3:ppmi
+0.075	+1:postag:DT	+0.086	-2:postag[:2]:CC			+0.072	+3:ppmi
+0.075	+1:postag[:2]:DT	+0.086	-2:postag:CC			+0.071	-1:ppmi
+0.066	+1:word.lower():the	+0.064	+1:word.lower():that			+0.065	-1:postag:NN
+0.064	+2:postag[:2]:NN	+0.060	word[-3:]:the			*… 9 more positive …*	
-0.021	+1:postag[:2]:NN	+0.060	word[-3:]:all			*… 11 more negative …*	
		+0.058	-1:word2vec			-0.060	+1:word.lower():all
		+0.048	word.lower():all			-0.099	-3:postag[:2]:PR
		+0.031	+1:postag[:2]:NN			-0.106	-3:postag[:2]:VB
		+0.014	word.lower():that			-0.108	-1:word.lower():and
		+0.013	postag:DT			-0.112	postag:NN
		+0.013	postag[:2]:DT			-0.133	postag:IN
		… 3 more positive …				-0.133	postag[:2]:IN
		-0.041	word.istitle()			-0.147	word.lower():and
		-0.044	+1:ppmi			-0.166	-1:postag:DT
		-0.047	+2:ppmi			-0.166	-1:postag[:2]:DT
		-0.059	postag[:2]:VB			-0.193	word[-3:]:out
		-0.092	-1:postag[:2]:NN			-0.199	word[-3:]:all
		-0.101	EOS			-0.229	word.lower():all
						-0.252	-1:word.lower():all

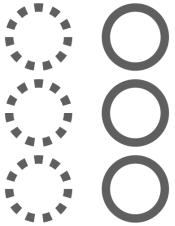


WHY IT FAILED

WHY IT FAILED

Lack of datasets

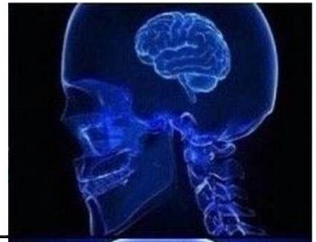
Pragmatics



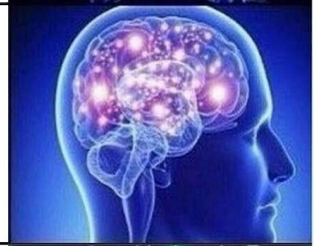
Similarity doesn’t seem to be a good indicator

POTENTIAL NEXT STEPS - TRANSLATION

Rule-based Translation 1960-1990

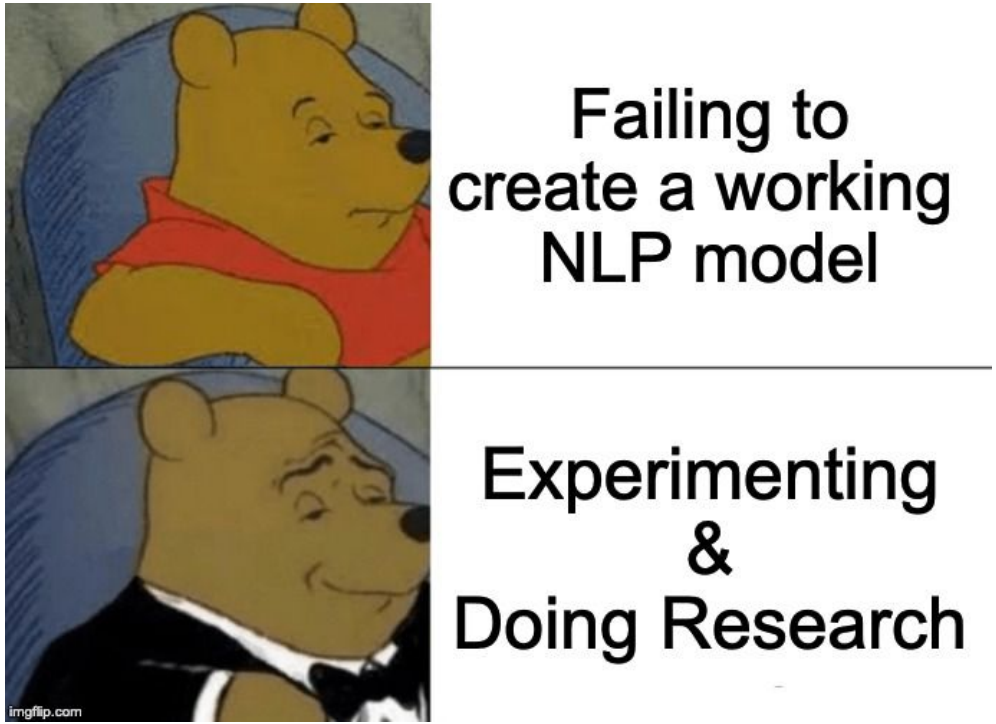


Statistical Translation 1990-2008



Neural Machine Translation 2008-

THE MORAL OF THE STORY IS...



FIRST QUESTION AWARD GOES TO