TechTalents
sólo es el principio

Natural Language Processing

Bringing the un-ordered human world to Python

A little background...

NLP is:

A little background...

NLP is:

- the endeavour to allow computers to understand/interact with humans, and vice-versa.

A little background...

NLP is:

- the endeavour to allow computers to understand/interact with humans, and vice-versa.

- simply the processes by which unstructured, “human”, language is converted into data that a computer can work with.

A little background...

NLP is:

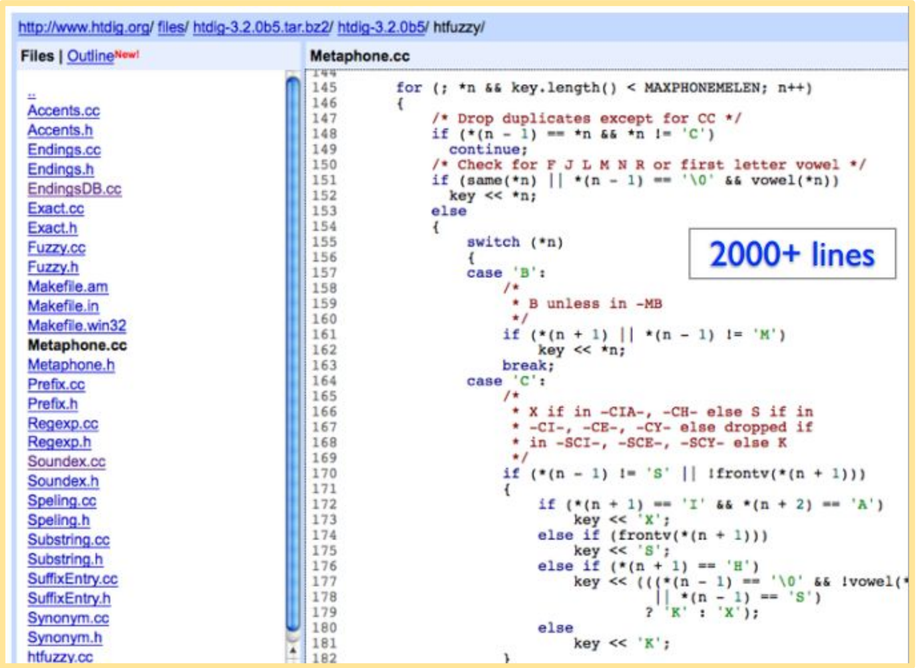
- the endeavour to allow computers to understand/interact with humans, and vice-versa.

- simply the processes by which unstructured, “human”, language is converted into data that a computer can work with.

- not new.

Unstructured -> Structured

Historically



http://www.htdig.org/ files/ htdig-3.2.0b5.tar.bz2/ htdig-3.2.0b5/ htfuzzy/

Files | Outline New!

..
Accents.cc
Accents.h
Endings.cc
Endings.h
EndingsDB.cc
Exact.cc
Exact.h
Fuzzy.cc
Fuzzy.h
Makefile.am
Makefile.in
Makefile.win32
Metaphone.cc
Metaphone.h
Prefix.cc
Prefix.h
Regexp.cc
Regexp.h
Soundex.cc
Soundex.h
Speling.cc
Speling.h
Substring.cc
Substring.h
SuffixEntry.cc
SuffixEntry.h
Synonym.cc
Synonym.h
htfuzzy.cc

Metaphone.cc

```
        for (; *n && key.length() < MAXPHONEMELEN; n++)
        {
            /* Drop duplicates except for CC */
            if (*(n - 1) == *n && *n != 'C')
              continue;
            /* Check for F J L M N R or first letter vowel */
            if (same(*n) || *(n - 1) == '\0' && vowel(*n))
              key << *n;
            else
            {
                switch (*n)
                {
                case 'B':
                    /*
                     * B unless in -MB
                     */
                    if (*(n + 1) || *(n - 1) != 'M')
                        key << *n;
                    break;
                case 'C':
                    /*
                     * X if in -CIA-, -CH- else S if in
                     * -CI-, -CE-, -CY- else dropped if
                     * in -SCI-, -SCE-, -SCY- else K
                     */
                    if (*(n - 1) != 'S' || !frontv(*(n + 1)))
                    {
                        if (*(n + 1) == 'I' && *(n + 2) == 'A')
                            key << 'X';
                        else if (frontv(*(n + 1)))
                            key << 'S';
                        else if (*(n + 1) == 'H')
                            key << (((*(n - 1) == '\0' && !vowel(*
                                     || *(n - 1) == 'S')
                                    ? 'K' : 'X');
                        else
                            key << 'K';
                    }
```

2000+ lines

Rule based.

Worked with pre-programmed rules, or “heuristics”.

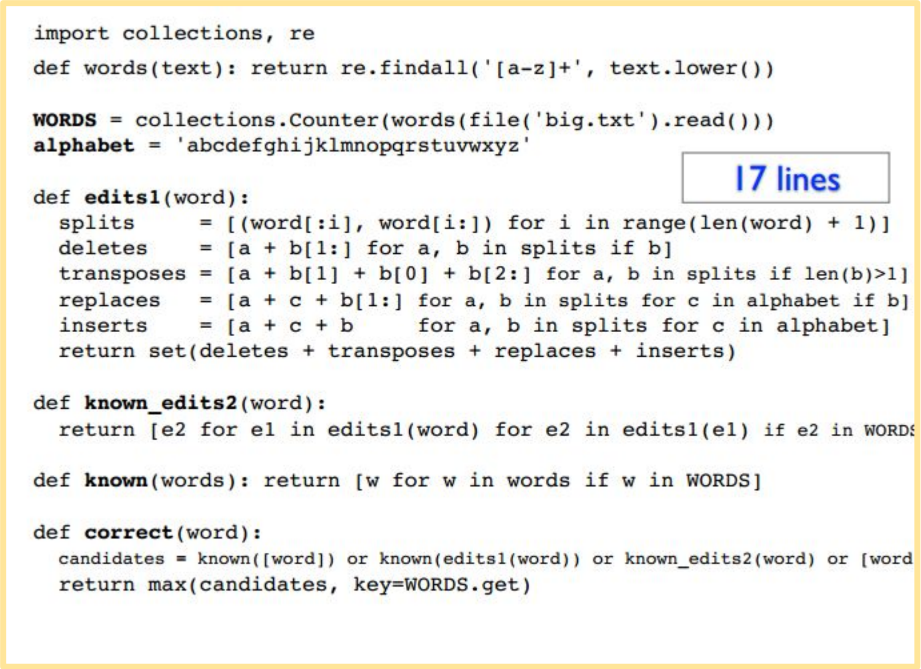
Basically a bunch of fancy if/else statements.

Modern NLP

Statistical approach.

Uses machine learning to infer the rules of grammar.

Works with large datasets to dynamically determine context and meaning.



```
import collections, re
def words(text): return re.findall('[a-z]+', text.lower())

WORDS = collections.Counter(words(file('big.txt').read()))
alphabet = 'abcdefghijklmnopqrstuvwxyz'

def edits1(word):
  splits     = [(word[:i], word[i:]) for i in range(len(word) + 1)]
  deletes    = [a + b[1:] for a, b in splits if b]
  transposes = [a + b[1] + b[0] + b[2:] for a, b in splits if len(b)>1]
  replaces   = [a + c + b[1:] for a, b in splits for c in alphabet if b]
  inserts    = [a + c + b     for a, b in splits for c in alphabet]
  return set(deletes + transposes + replaces + inserts)

def known_edits2(word):
  return [e2 for e1 in edits1(word) for e2 in edits1(e1) if e2 in WORDS

def known(words): return [w for w in words if w in WORDS]

def correct(word):
  candidates = known([word]) or known(edits1(word)) or known_edits2(word) or [word
  return max(candidates, key=WORDS.get)
```

17 lines

http://norvig.com/spell-correct.html

Some of the challenges:

Some of the challenges:

Capitalisation

“It was not great for us”

“It was not great for US”

“it was not great for us”

TechTalents
sólo es el principio

Some of the challenges:

Capitalisation

“It was not great for us”

“It was not great for US”

“it was not great for us”

Word Disambiguation

“The lost children were found by the searchers.”

“The lost children were found by the mountain.”

“The lost children were found by the afternoon.”

Some of the challenges:

Capitalisation

“It was not great for us”

“It was not great for US”

“it was not great for us”

Referents

“She killed the man with the tie.”

Word Disambiguation

“The lost children were found by the searchers.”

“The lost children were found by the mountain.”

“The lost children were found by the afternoon.”

“Loose” Basic Steps

1. Cleaning

   Remove unwanted, unneeded, unnecessary data.

2. Tokenization

   Split data pieces that fit the analysis being done.

3. Tagging

   Calculate and add “metadata”.

Next Steps

1. Normalization

   Retrieve the root/source of the word.

2. Contextualization

   Work out the meaning of words depending on their position.

3. Extraction

   Pull new understanding out of the data.

Tokenization

Splitting data into manageable and representational pieces.

Tokenization

Splitting data into manageable and representational pieces.

```
sentence = "python is wonderful and amazing and great"
sentence.split()
```

Tokenization

Splitting data into manageable and representational pieces.

```
~~sentence = "python is wonderful and amazing and great"~~
sentence = "Python's a great language for NLP. It is, isn't it?"
sentence.split()
```

Tokenization

Splitting data into manageable and representational pieces.

```
~~sentence = "python is wonderful and amazing and great"~~
sentence = "Python's a great language for NLP. It is, isn't it?"
sentence.split()
```

```
["Python's", 'a', 'great', 'language', 'for', 'NLP.', 'It', 'is,', "isn't", 'it?']
```

Tokenization

Splitting data into manageable and representational pieces.

```
~~sentence = "python is wonderful and amazing and great"~~
sentence = "Python's a great language for NLP. It is, isn't it?"
sentence.split()
```

```
["Python's", 'a', 'great', 'language', 'for', 'NLP.', 'It', 'is,', "isn't", 'it?']
```

Splitting into sentences.

Splitting into sentences.

```
text = "Hello there. How are you today? It's great weather."
```

Splitting into sentences.

```
text = "Hello there. How are you today? It's great weather."
```

Punctuation, Space, Capital?

Splitting into sentences.

```
text = "Hello Mr. Smith. How are you today? It's great weather."
```

Splitting into sentences.

```
text = "Hello Mr. Smith. How are you today? It's great weather."
```

Splitting into sentences.

```
text = "Hello Mr. Smith. How are you today? It's great weather."
```

Natural Language Toolkit to the rescue!

Normalization

Stemming

Removing and replacing of suffixes to get the root form of the word, called the **stem**.

Normalization

Stemming

Removing and replacing of suffixes to get the root form of the word, called the **stem**.

wolf, wolves -> wolf
talk, talks -> talk

Normalization

Lemmatization

Performing contextual and “morphological” analysis of the text, to determine closest correct root word.