English Word Suggestion Based on Part of Speech Ngram

Hamana Laboratory, Gunma University

Borann Chanrathnak

February 16, 2020

Table of Contents

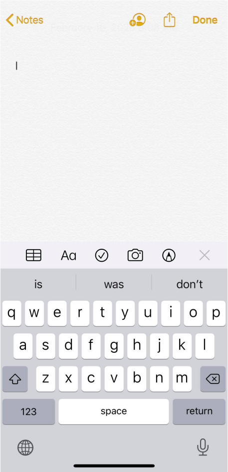
Inspiration

Inspiration

1. Word suggestion on mobile devices
2. Online digital writing tools

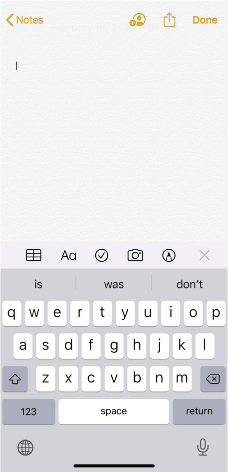
Inspiration

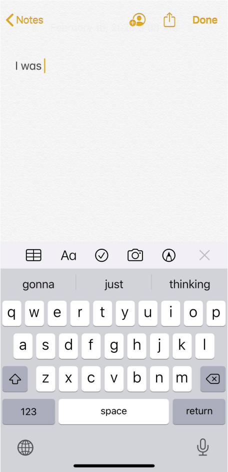
1. Word suggestion on mobile devices
2. Online digital writing tools



Inspiration

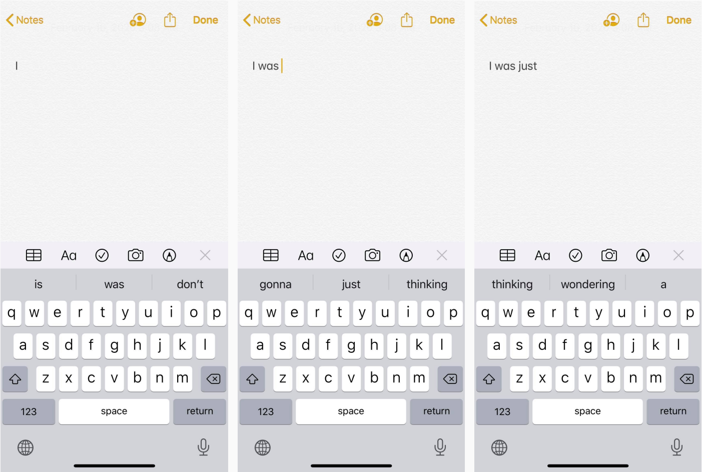
1. Word suggestion on mobile devices
2. Online digital writing tools





Inspiration

1. Word suggestion on mobile devices
2. Online digital writing tools



How to Predict?

"I am japanese, so I speak"

How to Predict?

"I am japanese, so I speak" $\rightarrow$

How to Predict?

"I am japanese, so I speak" $\rightarrow$ "japanese"

POS-Ngram

What does POS-Ngram mean?

Definition

Part of Speech (POS) one of the classes into which words are divided according to their grammar, such as noun, verb, adjective, etc.

Definition

N-gram is a contiguous sequence of n items from a given sample of text or speech

- **unigram (1-gram)** : (I,), (study,), (english,)
- **bigram (2-gram)** : (I, study), (study, english)
- **trigram (3-gram)** : (I, study, english)

What does POS-Ngram mean?

Definition

N-gram model is one of statistical langauge models for predicting the next item (word) based on Markov assumption, and usually abbreviated as N-gram.

Definition

POS-Ngram is an improved model using part of speech as a class indicator.

How To Compute Probability of a Sentence?

How can we compute the joint probability of a sentence?
Ex: "an apple is on the"

$$P(an\ apple\ is\ on\ the) = ?$$

Chain Rule Of Probability

Notation

Chain Rule Of Probability

Notation

- To represent the probability of a paricular random variable X_i taking on the value "the", or $P(X_i = "the")$ we will use simplification $P(the)$.

Chain Rule Of Probability

Notation

- To represent the probability of a paricular random variable X_i taking on the value "the", or $P(X_i = "the")$ we will use simplification $P(the)$.
- We represent a sequence of N words either as $w_1 \cdots w_n$ or w_1^n

Chain Rule Of Probability

In General

$$P(X_1 \cdots X_n) = P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_1^2) \cdots P(X_n \mid X_1^{n-1})$$

Chain Rule Of Probability

In General

$$P(X_1 \cdots X_n) = P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_1^2) \cdots P(X_n \mid X_1^{n-1})$$

By applying chain rule to a sequence of words

$$P(w_1 \cdots w_n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2) \cdots P(w_n \mid w_1^{n-1})$$

Chain Rule Of Probability

In General

$$P(X_1 \cdots X_n) = P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_1^2) \cdots P(X_n \mid X_1^{n-1})$$

By applying chain rule to a sequence of words

$$P(w_1 \cdots w_n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2) \cdots P(w_n \mid w_1^{n-1})$$

Examples

- $P(an\ apple) = P(an) \times P(apple \mid an)$
- $P(an\ apple\ is) = P(an) \times P(apple \mid an) \times P(is \mid an\ apple)$

Let's Apply Chain Rule

By applying chain rule to the phrase "*an apple is on the*":

$$
\begin{aligned}
P(\text{"}an\ apple\ is\ on\ the\text{"}) &= P(an) \times P(apple|an) \times P(is|an\ apple) \\
&\times P(on|an\ apple\ is) \times P(the|an\ apple\ is\ on)
\end{aligned}
$$

In practice chain rule does not help

- We don't know the way to compute the exact probability of a word given a long sequence of preceding words, $P(w_n|w_n^{n-1})$
- Language is creative, and any particular context might have never occured before

Example

$$P(\textit{sentences} \mid \textit{this is an example of long}) = ?$$

The Presence of N-gram

The general equation for this n-gram approximation to the conditional probability of the next word in a sequence is

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1})$$

The Presence of N-gram

The general equation for this n-gram approximation to the conditional probability of the next word in a sequence is

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1})$$

In case of Bigram (2-gram)

When we use bigram model to predict the conditional probability of the next word, we can approximate by

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1})$$

The Presence of N-gram

The general equation for this n-gram approximation to the conditional probability of the next word in a sequence is

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1})$$

In case of Bigram (2-gram)

When we use bigram model to predict the conditional probability of the next word, we can approximate by

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1})$$

Example

$$P(the|\ an\ apple\ is\ on) \approx P(the|\ on)$$

Bigram in Practice

By supposing that C is the counts of word from a corpus.

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)}$$

Bigram in Practice

By supposing that C is the counts of word from a corpus.

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)}$$

Example

$$P(the|\ on) = \frac{C(on\ the)}{\sum_{others} C(on\ others)}$$

The probability of a complete word sequence

Given the **bigram** assumption for the probability of an individual word, we can compute the probability of complete word sequence by:

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k \mid w_{k-1})$$

The probability of a complete word sequence

Given the **bigram** assumption for the probability of an individual word, we can compute the probability of complete word sequence by:

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k \mid w_{k-1})$$

Example

$$P(an\ apple\ is\ on\ the) = P(apple \mid an) \times P(is \mid apple) \times P(on \mid is) \\ \times P(on \mid is) \times P(the \mid on)$$

OOV & Smoothing

OOV stands for **Out Of Vocabulary**

- Words appear only in a test set but not in the training set.
- OOV problem occurs even when we work on big data.

Types of smoothing

- Laplace smoothing (Add-one smoothing)
- Add-k smoothing
- Interpolation
- $\cdots$

Interpolation

We mix the probability estimates from all the n-gram estimators, weighing and combining the trigram, bigram, and unigram counts.

Interpolation

To estimate $P(w_n \mid w_{n-2}w_{n-1})$ we use simple interpolation as follows:

Interpolation

To estimate $P(w_n \mid w_{n-2}w_{n-1})$ we use simple interpolation as follows:

In case of Bigram

$$\hat{P}(w_n \mid w_{n-1}) = \lambda_1 P(w_n \mid w_{n-1}) + \lambda_2 P(w_n)$$

Interpolation

To estimate $P(w_n \mid w_{n-2}w_{n-1})$ we use simple interpolation as follows:

In case of Bigram

$$\hat{P}(w_n \mid w_{n-1}) = \lambda_1 P(w_n \mid w_{n-1}) + \lambda_2 P(w_n)$$

In case of Trigram

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1 P(w_n|w_{n-2}w_{n-1}) \\ &+ \lambda_2 P(w_n|w_{n-1}) \\ &+ \lambda_3 P(w_n)\end{aligned}$$

Interpolation

To estimate $P(w_n \mid w_{n-2}w_{n-1})$ we use simple interpolation as follows:

In case of Bigram

$$\hat{P}(w_n \mid w_{n-1}) = \lambda_1 P(w_n \mid w_{n-1}) + \lambda_2 P(w_n)$$

In case of Trigram

$$\begin{aligned} \hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1 P(w_n|w_{n-2}w_{n-1}) \\ &+ \lambda_2 P(w_n|w_{n-1}) \\ &+ \lambda_3 P(w_n) \end{aligned}$$

where we choose λ_i such that $\sum_i \lambda_i = 1$

How are the λ values set?

- They can be learned from a **held-out** corpus
- Can be found by **EM** algorithm
- For the purpose of this project, we assume without loss of generality that $\lambda_i > \lambda_j (\forall i < j)$

POS-Ngram

Formula

$$P(w_n \mid w_1^{n-1}) = \sum_{c_n} P(w_n \mid c_n) \times P(c_n \mid c_{n-N+1}^{n-1})$$

POS-Ngram

Formula

$$P(w_n \mid w_1^{n-1}) = \sum_{c_n} P(w_n \mid c_n) \times P(c_n \mid c_{n-N+1}^{n-1})$$

Example

c_1:Noun	cat, dog, thought, $\cdots$
c_2:Verb	go, speak, $\cdots$
c_1:Noun, c_2:Verb	play, $\cdots$

POS-Ngram

Formula

$$P(w_n \mid w_1^{n-1}) = \sum_{c_n} P(w_n \mid c_n) \times P(c_n \mid c_{n-N+1}^{n-1})$$

Example

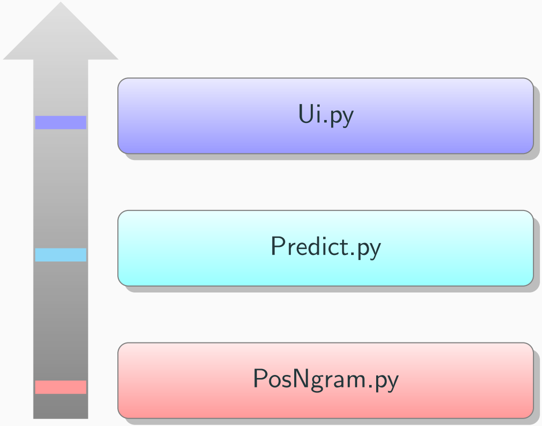
c_1:Noun	cat, dog, thought, $\cdots$
c_2:Verb	go, speak, $\cdots$
c_1:Noun, c_2:Verb	play, $\cdots$

It means that to predict the next word w_n

1. Compute $P(c_n|c_{n-N+1}^{n-1}) \sim P(w_n|w_{n-N+1}^{n-1})$
2. $P(w_n \mid c_n) = \frac{C(w_n, c_n)}{C(c_n)}$
3. w_n^* is defined by $\arg\max_{w_n} P(w_n \mid w_1^{n-1})$

Implementation

Programming Structure



Implementation

- Programming Language : Python
- Ui : Tkinter
- NLTK (Natural Language ToolKit)

Function	Argument	Result
word_tokenize	"I like english."	["I", "like", "english", "."]
pos_tag	["I", "like"]	[("I", "PRN"), ("like", "VBP")]

References

1. Speech and Language Processing (Chapter 4) (Daniel Jurafsky - Stanford University, 1999)
2. N-gram Language Modeling Tutorial (Lecture notes courtesy of Prof. Mari Ostendorf, 2007-06-21)
3. Probabilistic Language Model (Chapter 3) (Kenji KITA, University of Tokyo Press, 1999)

DEMO

Thank You.