Word2Vector

Speaker : 蘇佳益
Advisor : 陳聰毅
國立高雄科技大學建工校區電子工程系

https://github.com/chiayisu/Artificial_Intelligence_Course

Agenda

- Introduction
- N-Grams
- Introduction to Word Vector
- Complexity of Word Vector
- Negative Sampling
- Hierarchical Softmax
- Practical Topics
- Results
- References

Introduction

- N-Grams
  - Robustness
  - Simplicity
  - However: we cannot know similarity between words.
- WordVector (Mikolov et al.)
  - Represent words as dense vector
  - Able to perform simple algebraic operation
  - E.g.: Vector(queen) = Vector(king) – Vector(men) + Vector(women)

N-Grams

N-Grams: Introduction

- The simplest model to assign probabilities to each sentence
- Sequence of N words
- N can be any positive number. However: Usually (4-10)
- Our example will be N=2 (Bi-Gram)

N-Grams: Introduction

- Suppose we have a sentence, it is shown that, the word w is "that" given the history h "it is shown". Let's compute p(w|h).
- $p(w|h) = p(that|it\ is\ shown) = \frac{c(\text{it is shown that})}{c(\text{it is shown})}$

The computation above is problematic.

N-Grams: Why is it problematic?

- As we mentioned in “Introduction Course”, Natural Language often
  - evolves.
  - sometimes doesn’t follow grammar.
  - be equivocal.
- We may not find the sentences of the history in our database or web.

Thus, We need to consider few words

N-Grams: Bi-Grams

- According to Markov assumption
  - we only need to take few words into consideration.
  - Therefore: in previous example, we only need to calculate p(that | shown) in bi-grams model.

- Markov model is that we can predict the future words without looking too far into the past.

N-Grams: Bi-Grams - Estimation

- We estimate the probability intuitively by using Maximum Likelihood Estimation. The formula is as follows.
- $p(w_n|w_{n-1}) = \frac{C(W_{n-1}W_n)}{\sum_w C(W_{n-1}W)} = \frac{C(W_{n-1}W_n)}{C(W_{n-1})}$

Let's walk through an example.

N-Grams: Example

- Take a look at the following sentences
  - I am Ian
  - Ian am I
  - My name is Ian
- The following probabilities are the bi-grams probability in this corpus.
- P$(am\ |I) = \frac{1}{2}$
- P$(Ian\ |am) = \frac{1}{2}$
- P$(am|Ian\ ) = \frac{1}{3}$

Apparently, N-grams still make no sense about similarity.

Introduction to Word Vector

Previous Work

- Downside of Previous Model
  - computational costly
- E.g.: NNLM contains
  - feedforward neural network with a linear projection layer
  - non-linear hidden layer

Word2Vector Model (Mikolov et al.)

- Bag-of-Words model (CBOW)
- Continuous Skip-gram model

Complexity of Word Vector

Complexity Notation

- The following model’s complexity will be represented as follow.

➢ O = E * T * Q

- where E is the number of training epoch, usually 3 - 50, T is the number of words, usually 1billion and Q will be defined depending on the further model.

Complexity of NNLM

- NNLM is a language model that consists of input, projection, hidden and output layer. Given the N previous words, it will predict the next word.
- Complexity of Projection Layer
  - N * D, where N is the number of previous words and D is its dimension.
- Complexity of Hidden Layer
  - N * D * H, where N*D is from input layer and H is from Hidden layer.
- Complexity of Output Layer
  - H * V, where H is the number of hidden layer and V is the size of all vocabulary.
- Thus, its complexity, in total, is $Q = N * D + N * D * H + H * V$

CBOW

- CBOW Model
  - Is similar to NNLM
  - Predicts future words based on given history words
  - Is also similar to bag-of words model
  - However: heaviest layer hidden layer in NNLM is removed
  - Therefore: Its projection layer is shared with all words
- Complexity of CBOW
  - ➢ Q = N * D + D * V

Skip-Gram

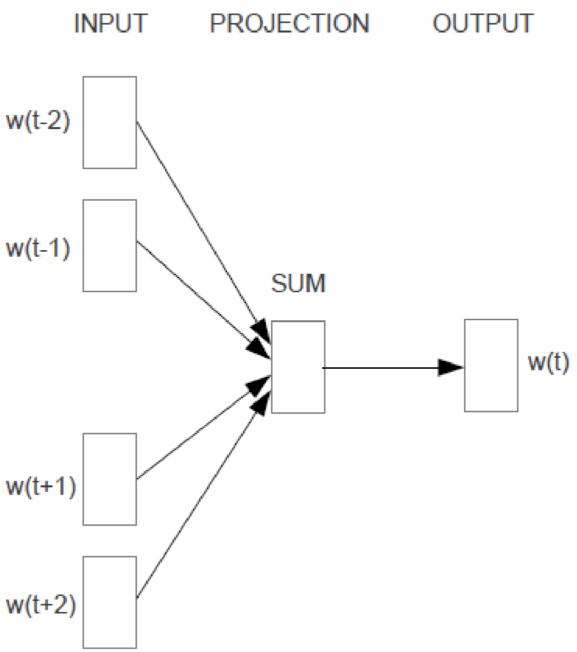
- Skip-Gram model
  - Is opposite from the CBOW.
  - Predicts the context based on given word
- Complexity of Skip-Gram
  - ➢ Q = C * (D + D * V), where C is the distance of the word. The size of C is double because it consists of size from history and future.

Complexity: Binary Tree Representation

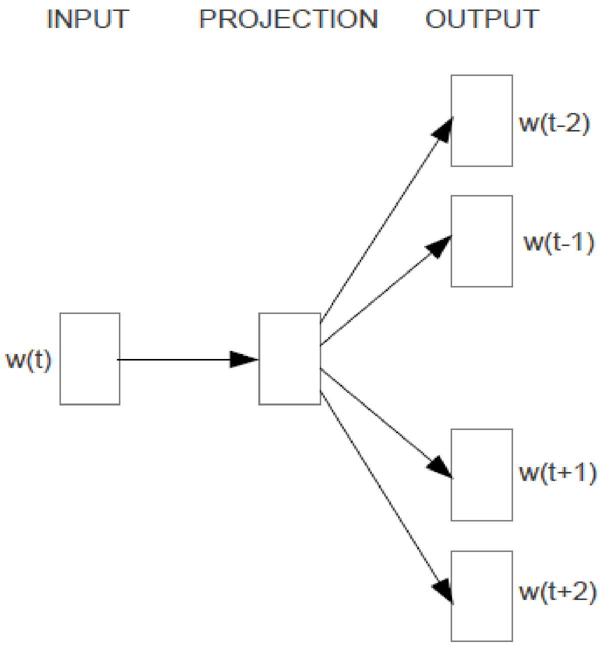
- Complexity of Binary Tree
  - log(n)
- Therefore: the complexity of representing vocabulary as binary tree is log(v)
- However: hierarchical Softmax is required instead of Softmax.

Algorithm of CBOW and Skip-Gram

Architecture of CBOW and Skip-Gram



CBOW



Skip-gram

CBOW

Notation for Algorithm

- w_i : $word\ i\ from\ vocabulary\ V$
- $\omega \in\ R^{n*|V|}$: $input\ word\ matrix$
- ω_i : $i - th\ coulumn\ of\ \omega$
- $U \epsilon R^{|V|*n}$: $ouput\ word\ matrix$
- u_i : $i - th\ coulumn\ of\ U$

Algorithm of CBOW

1. Generate one hot word vector for window size m :
   $\left(x^{(c-m)}, \ldots, x^{(c-1)}, \ldots, x^{(c+m)} \in R^{|V|}\right)$
2. Get embedded word vector($\hat{v} = \omega x^{(c-m)} + \cdots + \omega x^{(c+m)} \in R^{n}$)
3. Average it to get $\hat{v} = \frac{\hat{v}}{2m} \in R^{n}$
4. Generate score vector z =$U * \hat{v} \in R^{|V|}$
5. Turn the score into probability y = softmax(z) ∈ $R^{|V|}$
6. Run gradient descent to update input and output word matrix

Update Input and Output Word Matrix

- Information theory is employed when
  - we want to learn the probability from true probability.
- Cross-Entropy loss is derived as follows.
- $H(\hat{y}, y) = -y_i \log(\hat{y}_i)$
- Take a look at example, if $\hat{y}_i$ = 1, our H will be 0, which is no loss. If $\hat{y}_i$ = 0.01, our H will be ≈4.605

Update Input and Output Word Matrix (CONT.)

- Thus, we need to minimize our cross-entropy.
- Let J = $-\log P(w_c|w_{c-m},\dots,w_{c+m})$

$$= -logP(u_c|\hat{v})$$

$$= -log\frac{\exp(u_c^T\hat{v})}{\sum_{j=1}^{|V|}\exp\left(u_j^T\hat{v}\right)}$$

$$=-u_c^T\hat{v} + \log\left(\sum_{j=1}^{|V|}\exp\left(u_j^T\hat{v}\right)\right)$$

Gradient descent is required to update $\hat{v}$ and u_c

Skip-Gram

Algorithm of Skip-Gram

- Generate one hot input vector x ∈ $R^{|V|}$ of center word
- Get embedded word vector for center word$v_c = \omega x \in R^n$
- Generate a score vector z =$U * v_c$
- Turn score vector into probability $\hat{y} = softmax(z)$
- Update input and output word matrix

Algorithm of Skip-Gram (cont.)

- Basically, it's similar to CBOW. Input and Output matrices are required to update
- However: Skip-Gram assumes output words are completely independent as Naïve Bayes algorithm.
- Let J = $-\log P(w_{c-m}, \ldots, w_{c+m} | w_c)$

$$= -\log\left(\prod_{j=0, j\neq m}^{2m} P\left(w_{c-m+j} \middle| w_c\right)\right.$$

$$= -\log\left(\prod_{j=0, j\neq m}^{2m} P\left(u_{c-m+j} \middle| v_c\right)\right.$$

$$= -\log\left(\prod_{j=0, j\neq m}^{2m} \frac{\exp\left(u_{c-m+j}^T v_c\right)}{\sum_{k=1}^{|V|} \exp\left(u_k^T v_c\right)}\right)$$

$$= -\sum_{j=0, j\neq m}^{2m} u_{c-m+j}^T v_c + 2m\log\left(\sum_{k=1}^{|V|} \exp(u_k^T v_c)\right)$$

國立高雄科技大學
National Kaohsiung University of Science and Technology

Negative Sampling

Introduction

- Disadvantage of Softmax
  - Computational Costly
  - Reason: takes $O(|V|)$ time
- Negative Sampling
  - Only needs to loop over noise distribution
  - Is based on skip-gram model

Objective Function

- Let's denote (w, c) to word and context, $P(D = 1|w, c)$ to the probability that (w, c) comes from corpus data, and $P(D = 0|w, c)$ vice versa. Therefore, we can model $P(D = 1|w, c)$ with sigmoid function as follows.
- $P(D = 1|w, c, \theta) = \sigma(v_c^T v_w) = \frac{1}{1+e^{-(v_c^T v_w)}}$

Objective Function (CONT.)

- New objective function is built to maximize the probability of a word and context in the corpus data. It's derived bellow. We also denote θ as the parameters of input and output matrix.
- $\theta = \underset{\theta}{\operatorname{argmax}} \prod_{(w,c)\in D} p(D=1|w,c,\theta) \prod_{(w,c)\in \hat{D}} p(D=0|w,c,\theta)$

$= \underset{\theta}{\operatorname{argmax}} \prod_{(w,c)\in D} p(D=1|w,c,\theta) \prod_{(w,c)\in \hat{D}} (1-p(D=1|w,c,\theta))$

$= \underset{\theta}{\operatorname{argmax}} \sum_{(w,c)\in D} \log(p(D=1|w,c,\theta)) + \sum_{(w,c)\in \hat{D}} \log(1-p(D=1|w,c,\theta))$

$= \underset{\theta}{\operatorname{argmax}} \sum_{(w,c)\in D} \log(\frac{1}{1+\exp(-u_w^T v_c)}) + \sum_{(w,c)\in \hat{D}} \log(1-\frac{1}{1+\exp(-u_w^T v_c)})$

$= \underset{\theta}{\operatorname{argmax}} \sum_{(w,c)\in D} \log(\frac{1}{1+\exp(-u_w^T v_c)}) + \sum_{(w,c)\in \hat{D}} \log(\frac{1}{1+\exp(u_w^T v_c)})$

Objective Function (CONT.)

- Because maximizing log likelihood is the same as minimizing negative log likelihood, our function will be revised bellow.
- $\mathrm{J} = -\sum_{(w,c)\in D} \log(\frac{1}{1+\exp(-u_w^T v_c)}) - \sum_{(w,c)\in \widehat{D}} \log(\frac{1}{1+\exp(u_w^T v_c)})$
  $\widehat{D}\ is\ a\ false\ corpus,\ where\ unnatural\ sentenses\ contain\ in\ this\ corpus.$ $\widehat{D}$ can be generated by randomly sampling from word bank.

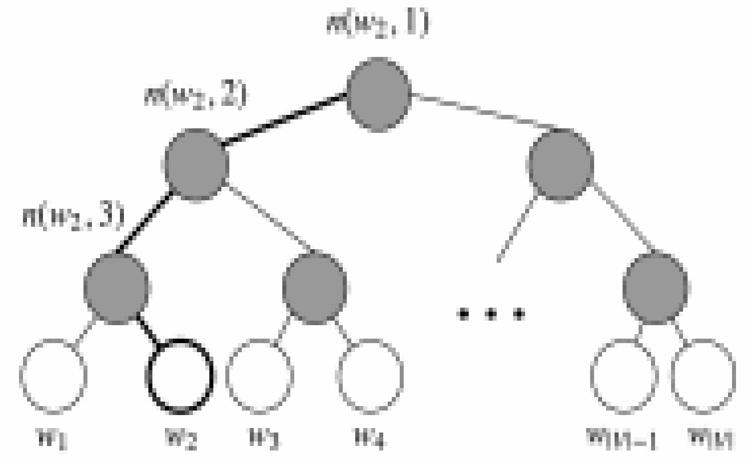
Objective Function (CONT.)

- New objective function for skip-gram model is, thus, as follows
- $-\log(\sigma\left(u_{c-m+j}^{T} * v_c\right) - \sum_{k=1}^{K} \log(\sigma(-\hat{u}_k^T * v_c)$
- New objective function for CBOW model is, thus, as follows
- $-\log(\sigma(u_c^T * v_c) - \sum_{k=1}^{K} \log(\sigma(-\hat{u}_k^T * v_c)$
- $\{\hat{u}_k | k = 1 \ldots K\}$ are sampled from P(w). P(W) is generally unigram model raise to $\frac{3}{4}$ because the unnatural words are more likely to be sampled.

Hierarchical Softmax

Introduction

- Hierarchical softmx
  - represents all words as binary tree.
  - Each leaf is a word and each node is a vector that model is going to learn.
- Advantage of using hierarchical softmax
  - Only cost O(log(V))
- Disadvantage of using hierarchical softmax
  - Cannot be parallelized



Notation

- L(w) : number of node from root to leaf w
- n(w,i) : i-th node on the path
- ch(i) : inner node n’s children

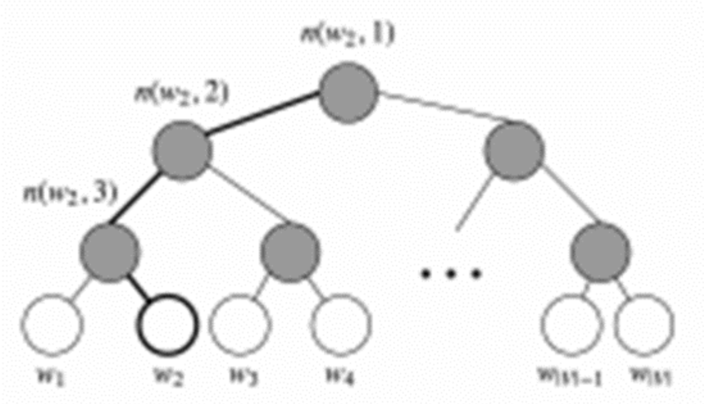
Probability Calculation

- The probability is, then, as follows.
- $P(w|w_i) = \prod_{j=1}^{L(w)} \sigma\left([n(w, j+1) = ch(n(w,j))] * v_{n(w,j)}^{T} v_{w_i}\right)$
- $where\ [x] = \begin{cases} 1\ if\ x\ is\ true \\ -1\ otherwise \end{cases}$
- We also assume that ch(n) is the left node of n. if path goes left, $[n(w, j+1) =$

Example

- take the graph bellow for example. The approach to calculate $p(w_2|w_i)$ is as follows.
- $p(w_2|w_i) = p(n(w_2,1), left) * p(n(w_2,2), left) * p(n(w_2,3), right)$

$$= \sigma\left(v_{n(w_2,1)}^{T} v_{w_i}\right) * \sigma\left(v_{n(w_2,2)}^{T} v_{w_i}\right) * \sigma\left(-v_{n(w_2,3)}^{T} v_{w_i}\right)$$



Our goal is the same as before, which is minimize $-\log p(w|w_i)$

Practical Topics

Subsampling of Frequent Words

- Meaningless words
  - For instance: a, an, the
  - are often in the large corpora
  - Appear most of time
- However: Skip-gram model doesn't benefit from this kind of words
- Therefore: we need to counter the imbalance.
- P$(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$, $f(w_i)$ is the frequency of the word and is a chosen threshold. It is typically around 10^{-5}.

Learning Phrase

- Some words appear together as phrases
  - E.g. New York Times
- This paper (Mikolov et al.) only uses simple model based on unigram and bigram counts.
- $score(w_i, w_j) = \frac{count(w_i, w_j) - \delta}{count(w_i) * count(w_j)}$
- δ is a coefficient to avoid too many infrequent phrases. We choose the phrases that have the score above the threshold.
- This algorithm is often run 2-4 passes with declining threshold.

Results

Examples of Semantic and Syntactic Test

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Accuracy on Semantic-Syntactic Word Relationship Test Set with CBOW

Dimensionality / Training words	24M	49M	98M	196M	391M	783M
50	13.4	15.7	18.6	19.1	22.5	23.2
100	19.4	23.1	27.8	28.7	33.4	32.2
300	23.2	29.2	35.3	38.6	43.7	45.9
600	24.0	30.1	36.5	40.8	46.6	50.4

- The much data and higher dimension we use to train model, The higher accuracy we get.

Comparison of Model Trained on Same Data, with 640-Dimensional Word Vectors

Model Architecture	Semantic-Syntactic Word Relationship test set	
	Semantic Accuracy [%]	Syntactic Accuracy [%]
RNNLM	9	36
NNLM	23	53
CBOW	24	$\boxed{64}$
Skip-gram	$\boxed{55}$	59

Comparison of Models Using Distbelied Distributed Framework

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days x CPU cores]
			Semantic	Syntactic	Total	
NNLM	100	6B	34.2	64.5	50.8	14 x 180
CBOW	1000	6B	57.3	68.9	63.7	2 x 140
Skip-gram	1000	6B	66.1	65.1	65.6	2.5 x 125

- It takes too long to finishing training 1000 dimension on NNLM model.

Example of Word Pair Relationships Using Best Skip-Gram Model

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

- According to paper (Tomas Mikolov et al.), it only reaches 60% on accuracy, but it will be improved if we train on larger data sets.

Accuracy of Various 300-Dimensional Skip-Gram Model

Method	Time [min]	Syntactic [%]	Semantic [%]	Total accuracy [%]
NEG-5	38	63	54	59
NEG-15	97	63	58	**61**
HS-Huffman	41	53	40	47
NCE-5	38	60	45	53
The following results use 10^{-5} subsampling				
NEG-5	14	61	58	60
NEG-15	36	61	61	**61**
HS-Huffman	21	52	59	55

- NEG-k means Negative Sampling with k negative samples.

Accuracy of Skip-Gram Models on the Phrase Analogy Dataset

Method	Dimensionality	No subsampling [%]	10^{-5} subsampling [%]
NEG-5	300	24	27
NEG-15	300	27	42
HS-Huffman	300	$\boxed{19}$	$\boxed{\mathbf{47}}$

	NEG-15 with 10^{-5} subsampling	HS with 10^{-5} subsampling
Vasco de Gama	Lingsugur	Italian explorer
Lake Baikal	Great Rift Valley	Aral Sea
Alan Bean	Rebbeca Naomi	moonwalker
Ionian Sea	Ruegen	Ionian Islands
chess master	chess grandmaster	Garry Kasparov

- we got the lowest accuracy on HS-Huffman without subsampling, but we reached highest accuracy when we trained with subsampling.
- This result reached 72% of accuracy.

Comparison of Model with Previous Models

Model (training time)	Redmond	Havel	ninjutsu	graffiti	capitulate
Collobert (50d) (2 months)	conyers lubbock keene	plauen dzerzhinsky osterreich	reiki kohona karate	cheesecake gossip dioramas	abdicate accede rearm
Turian (200d) (few weeks)	McCarthy Alston Cousins	Jewell Arzu Ovitz	- - -	gunfire emotion impunity	- - -
Mnih (100d) (7 days)	Podhurst Harlang Agarwal	Pontiff Pinochet Rodionov	- - -	anaesthetics monkeys Jews	Mavericks planning hesitated
Skip-Phrase (1000d, 1 day)	Redmond Wash. Redmond Washington Microsoft	Vaclav Havel president Vaclav Havel Velvet Revolution	ninja martial arts swordsmanship	spray paint grafitti taggers	capitulation capitulated capitulating

- Empty means words was not in the vocabulary sets.

References

- Mikolov et al., Efficient Estimation of Word Representations in Vector Space
- Mikolov et al., Distributed Representation of Words and Phrases and their Compositionality
- Manning et al., CS224N WordVector Lecture Note in Stanford University
- Alex Minnaar Word2Vec Tutorial Part II: The Continuous Bag-of-Words Model
- Jurafsky & Martin, Speech and Language Processing N-grams