Lecture 3

Andrei Arsene Simion

Columbia University

February 1, 2023

Outline

- Glove
- fastText
- Skip Gram with Negative Sampling is Matrix Factorization

Outline

- Glove
- fastText
- Skip Gram with Negative Sampling is Matrix Factorization

Background

- The models we saw, CBOW, Skip-Gram, the first neural language model (Lecture 1), were all based on prediction
- They are "online" methods: you get a subset (batch) of data, and you update the model based on this data
- You predict one word from a context, and as a result you get word vectors which are "good"
- Can you do well if you use global "counts" in some way?
- Can we optimize a global objective directly and get good vectors?

Background

- Let V be some vocabulary with words $\{w_1, \ldots, w_{|V|}\}$
- Let X be a co-occourence matrix with X_{ij} meaning that word j occurs in the context of word i
  - X **need not** be a count, but we ca say $X_i j \sim N(w_i, w_j)$, the count of w_i and w_j being in the same context
- **Task**: *Unlike local models like Skip-Gram or CBOW, we want to get the word embeddings from "global" properties as captured by X*
- Note the standard notation
  - Let $X_i = \sum_k X_{ik}$ be the number of times any word k occurs in the context of word i
  - Let $P_{ij} = P(j|i) = \frac{X_{ij}}{X_i}$ be the probability that word j occurs in the context of word i
  - **Note**: We'll use i and w_i interchangably to refer to the word w_i that has index i in V

Intuition

- Let $w_i = ice$ and $w_j = steam$
- Consider $F(i, j, k) = \frac{P_{ik}}{P_{jk}}$ for various other words k
  - When $i \sim k$, $j \not\sim k$ ($k = solid$) we expect $F(i, j, k)$ to be

$$LARGE = \frac{P(solid|ice)}{P(solid|stream)} \sim 0.97/0.01$$

  - When $i \not\sim k$, $j \sim k$ ($k = gas$) we expect $F(i, j, k)$ to be

$$SMALL = \frac{P(gas|ice)}{P(gas|stream)} \sim 0.0/0.95$$

  - When $i \sim k$, $j \sim k$ ($k = water$) we expect $F(i, j, k)$ to be

$$1 = \frac{P(water|ice)}{P(water|stream)} \sim 0.95/0.92$$

  - When $i \not\sim k$, $j \not\sim k$ ($k = fashion$) we expect $F(i, j, k)$ to be

$$1 = \frac{P(fashion|ice)}{P(fashion|stream)} \sim 0.002/0.003$$

Intuition

Table 1: Co-occurrence probabilities for target words *ice* and *steam* with selected context words from a 6 billion token corpus. Only in the ratio does noise from non-discriminative words like *water* and *fashion* cancel out, so that large values (much greater than 1) correlate well with properties specific to ice, and small values (much less than 1) correlate well with properties specific of steam.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

- **Key** : compared to the raw probabilities, the ratio $F(i, j, k)$ is better able to distinguish relevant words (solid and gas) from irrelevant words (water and fashion) and it is also better able to discriminate between the two relevant words

Model

- If we want to learn vectors by co-occourence, we should consider this ratio
- Generally, let $F(a_i, a_j, b_k) = \frac{P_{ik}}{P_{jk}}$
  - $a \in \mathbb{R}^d$ is a word vector
  - $b \in \mathbb{R}^d$ is a separate context vector
- **Question**: How can we put some structure on F?
  - We want F to encode the information in the ratio $\frac{P_{ik}}{P_{jk}}$
  - Restrict F to only depend on the difference of i and j so that we have $F(a_i - a_j, b_k) = \frac{P_{ik}}{P_{jk}}$

Model

- $F(a_i - a_j, b_k) = \frac{P_{ik}}{P_{jk}}$ is still very complicated so we simplify further
- $F((a_i - a_j)^\intercal b_k) = \frac{P_{ik}}{P_{jk}}$ brings in the inner product
- **Note** : context is arbitrary for co-occourence, and we can exchange the two roles, we can swap $a \leftrightarrow b$ and $X \leftrightarrow X^\intercal$
  - We want the above to hold true for our model, but it does not - get there in 2 steps
  - Step 1: simply further to allow $F((a_i - a_j)^\intercal b_k) = \frac{F(a_i^\intercal b_k)}{F(a_j^\intercal b_k)}$

Model

- $F((a_i - b_j)^\mathsf{T} b_k) = \frac{F(a_i^\mathsf{T} b_k)}{F(a_j^\mathsf{T} b_k)}$
- The above looks like $F(x - y) = \frac{F(x)}{F(y)}$
- **A solution to the above is** $F(x) = \exp x$
- We have $\frac{F(a_i^\mathsf{T} b_k)}{F(a_j^\mathsf{T} b_k)} = \frac{P_{ik}}{P_{jk}}$
- But also $F(a_i^\mathsf{T} b_k) = P_{ik}$
- $P_{ik} = \frac{X_{ik}}{X_i}$
- Since $F(a_i^\mathsf{T} b_k) = \exp a_i^\mathsf{T} b_k$We then have a solution if $a_i^\mathsf{T} b_k = \log(X_{ik}) - \log(X_i)$
- Step 2: this still does not have the exchange symmetry $i \leftrightarrow k$; it does have this symmetry if we assume more that

$$a_i^\mathsf{T} b_k + c_i + d_k = \log(X_{ik})$$

Model

- Given the above, we can try and optimize

$$J = \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} (a_{w_i}^{\mathsf{T}} b_{w_j} + c_{w_i} + d_{w_j} - \log(X_{ij}))^2$$

- Parameters: A, B, c, d where A, B are $|V| \times d$ matrices and c, d are vectors of dimension $|V|$
- This gives the same answer if we swap $a \leftrightarrow b$ and $X \leftrightarrow X^{\mathsf{T}}$
- This has issues
  - What if $X_{ij} = 0$?
  - What if (i, j) are rare or (i, j) are frequent? All pairs have the same weight, but rare co-occourence pairs are much noisier

Model

- Given the above, we can try and optimize

$$J = \sum_{i,j}^{|V|} f(X_{ij})(a_i^\intercal b_j + c_i + d_j - \log(X_{ij}))^2$$

- We have f to have some nice properties
  - $f(0) = 0$ and $\lim_{x \to 0} f(x) \log^2(x) = 0$ for stability
  - $f(x)$ should be non-decreasing, so that rare co-occourence are not overweighted
  - $f(x)$ should be relatively small for large values of x, so that frequent co-occourence are not overweighted

Model

- The authors choose f as below

$$f(x) = \begin{cases} (\frac{x}{x_{max}})^{\alpha} & \text{if } x \leq x_{max} \\ 1 & \text{otherwise} \end{cases}$$

- Note that this has the properties we need if we use for example $\alpha = 3/4$ and $x_{max} = 100$
- Authors also used $\alpha = 1$ but $\alpha = 3/4$ was the best in experiments

Model

- GloVe objective is

$$J = \sum_{i,j}^{|V|} f(X_{ij})(a_i^\mathsf{T} b_j + c_i + d_j - \log(X_{ij}))^2$$

- Authors note that the objective is $O(|V|^2)$: Is this a problem?
  - Shallow window approaches scale like $O(W|C|)$ where C is the corpus and W is the window size
- Assume that $X_{ij} \sim \frac{k}{r_{ij}^\beta}$ where r_{ij} is the (integer) frequency rank of the pair (i, j)
  - Note that the highest value of r is $|X|$
  - This is the largest r such that $\frac{k}{r^\beta} \geq 1$

Model

- Reminder: the corpus size is such that the size $|C|$ is proportional to the sum of X, $|C| \sim \sum_{i,j} X_{ij} = \sum_{r=1}^{|X|} \frac{k}{r^{\beta}}$
  - Question: **Why is** $|C| \sim \sum_{i,j} X_{ij}$?
  - Think about walking the corpus
- The maximum value of r_{ij} such that $X_{ij} \geq 1$ is $|X|$ so then $|X| = k^{1/\beta}$
- The authors show that under this distribution of X_{ij} and for their data $|X| = O(|C|^{1/\beta})$ if $\beta > 1$ and $O(|C|)$ otherwise
- So, they in fact show that the complexity of J is of the type $O(|C|^{0.8})$ which is much better than $O(|V|^2)$ and like Skip-Gram / CBOW

Connections with Online Models

- Skip-Gram can be written as minimizing

$$J = - \sum_{w_i \in C, w_j \in context(w_i)} \log P(w_j|w_i)$$

- Let $Q_{ij} = \log P(w_j|w_i)$ where we associate with each word a unique integer
- The above can be written in another way as

$$J = - \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} X_{ij} \log Q_{ij}$$

- In this case, $X_{ij} = N(w_i, w_j)$ is the number of times i and j are seen in the same context of each other

Connections with Online Models

- Note that $X_i = \sum_k X_{ik}$ so that the above is

$$J = -\sum_{i=1}^{|V|} X_i \sum_{j=1}^{|V|} P_{ij} \log Q_{ij} = \sum_{i=1}^{|V|} X_i \sum_{j=1}^{|V|} Cross - Entropy(P_i, Q_i)$$

- Issues with the above include
  - Cross-Entropy has the problem that distributions with long tails are modeled poorly and too much weight is given to rare events
  - Q_{ij} needs to be normalized and this is a problem since the denominator in the softmax is expensive to get, as we saw

Connections with Online Models

- We can try

$$J = -\sum_{i=1}^{|V|} X_i \sum_{j=1}^{|V|} (\hat{P}_{ij} - \hat{Q}_{ij})^2$$

- In the above, $\hat{P}_{ij} = X_{ij}$ and $\hat{Q}_{ij} = \exp a_i^\mathsf{T} b_j$
- Issues with the above include
  - X_{ij} can be very large and this complicates optimization

Connections with Online Models

- Attempt

$$J = -\sum_{i=1}^{|V|} X_i \sum_{j=1}^{|V|} (\log \hat{P}_{ij} - \log \hat{Q}_{ij})^2 = -\sum_{i=1}^{|V|} X_i \sum_{j=1}^{|V|} (a_i^\mathsf{T} b_j - \log X_{ij})^2$$

- Other issues with the above include
  - Frequent words get too much weight
- We can try

$$J = -\sum_{i,j} f(X_{ij})(a_i^\mathsf{T} b_j - \log X_{ij})^2$$

- The above is GloVe without the terms c_i and d_j

Implementation details?

- Authors walk the corpus C and update X for each word pair (i, j) with $X_{ij} += \frac{1}{d}$, where d is the distance between w_i and w_j
- They look at symmetric and asymmetric context
- The dimension of most GloVe vectors is about 300, as this seems to be the sweet spot

Experiments: How do you evaluate word vectors?

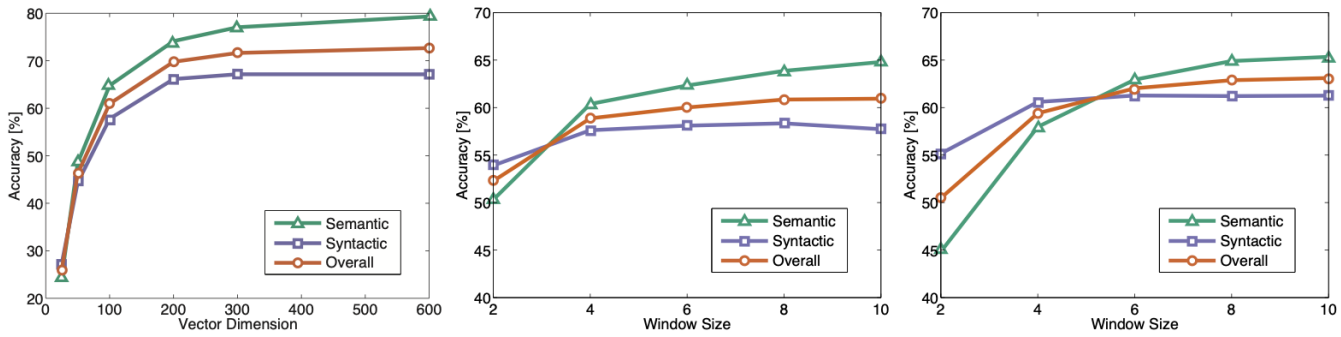
- Typical method used to say a word vector makes sense is analogies: "a is to b as c is to __"
- Semantic: "Athens is to Greece as Berlin is to __?"
- Synthetic "dance is to dancing as fly is to __?"
- Given w_a, w_b and w_c find d such that w_d is closest to $w_b - w_a + w_c$
- These are **intrinsic** evaluations: focus on a particular feature of the produced word vectors

Experiments: word analogies as intrinsic evaluation

Table 2: Results on the word analogy task, given as percent accuracy. Underlined scores are best within groups of similarly-sized models; bold scores are best overall. HPCA vectors are publicly available[2]; (i)vLBL results are from (Mnih et al., 2013); skip-gram (SG) and CBOW results are from (Mikolov et al., 2013a,b); we trained SG† and CBOW† using the `word2vec` tool[3]. See text for details and a description of the SVD models.

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	67.4	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	**81.9**	**69.3**	**75.0**

Experiments: word analogies

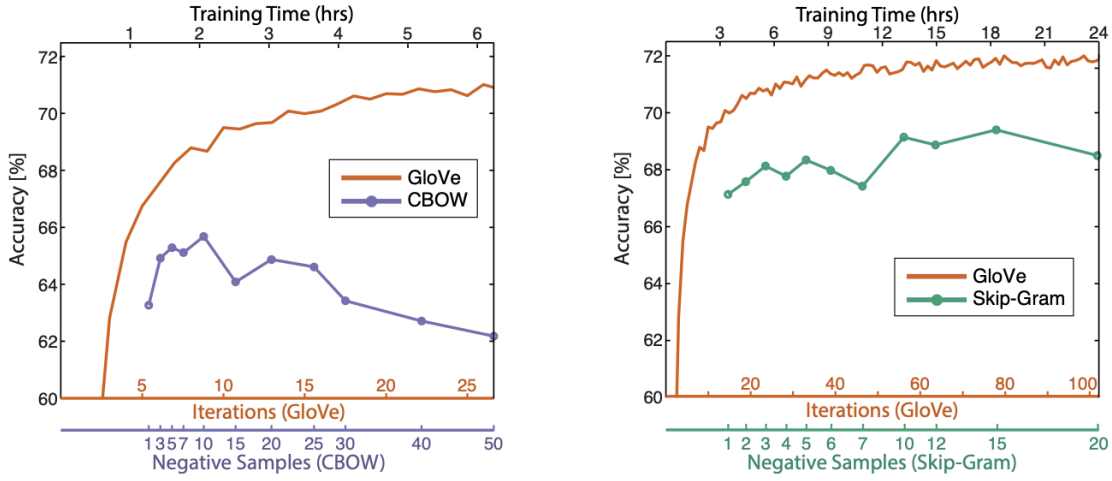


(a) Symmetric context (b) Symmetric context (c) Asymmetric context

Figure 2: Accuracy on the analogy task as function of vector size and window size/type. All models are trained on the 6 billion token corpus. In (a), the window size is 10. In (b) and (c), the vector size is 100.

- Vector dimension 300 seems ideal
- Context window 8 - 10 seems good
- Symmetric is better?

Experiments: word analogies



(a) GloVe vs CBOW (b) GloVe vs Skip-Gram

Figure 4: Overall accuracy on the word analogy task as a function of training time, which is governed by the number of iterations for GloVe and by the number of negative samples for CBOW (a) and skip-gram (b). In all cases, we train 300-dimensional vectors on the same 6B token corpus (Wikipedia 2014 + Gigaword 5) with the same 400,000 word vocabulary, and use a symmetric context window of size 10.

- Better than Skip-Gram or CBOW?
- Not really, they used better data

Experiments: extrinsic evaluation

Table 4: F1 score on NER task with 50d vectors. *Discrete* is the baseline without word vectors. We use publicly-available vectors for HPCA, HSMN, and CW. See text for details.

Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	**88.7**	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	**93.2**	88.3	**82.9**	**82.2**

shown for neural vectors in (Turian et al., 2010).

- **Extrinsic** evaluation: feed the word vectors into a complex system and see if the system's performance goes up
- NER task: identify organizations, people, etc in a dataset
- Example sentence: "EU rejects German call to boycott British lamb ."
- Example labels: ['B-ORG', 'O', 'B-MISC', 'O', 'O', 'O', 'B-MISC', 'O', 'O']

Outline

- Glove
- fastText
- Skip Gram with Negative Sampling is Matrix Factorization

fastText: Review of Skip-Gram

- At a high level, Skip-Gram aims to predict for each center word the words around it
- The objective is to maximize

$$\sum_{w_c \in C, w_o \in context(w_c)} \log P(w_o|w_c)$$

- $context(w_c)$ is the set of indices of words surrounding w_c, they make up the context of c

fastText

- Generally, we have that

$$P(w_o|w_c) = \frac{\exp s(b_{w_o}, a_{w_c})}{\sum_{j=1}^{|V|} \exp s(b_{w_j}, a_{w_c})}$$

- Here, we have $s(b_{w_o}, a_{w_c}) = b_{w_o}^{\mathsf{T}} a_{w_c}$

fastText: Negative sampling

- Typically, we replace the softmax by a negative sampling proxy so the term $\log P(w_o|w_c)$ is approximated by

$$\log \sigma(b_{w_o}^{\mathsf{T}} a_{w_c}) + \sum_{k=1}^{K} E_{w_k \sim P(w)}[\log \sigma(-b_{w_k}^{\mathsf{T}} a_{w_c})]]$$

- The negative samples are chosen randomly: they are, on average, not related to w_c
- $\sigma(x) = \frac{1}{1+e^{-x}}$

fastText

- Some problems with Skip-Gram
  - Ignores internal structure of words: each words represents a bag of characters
  - What do you do about words that you've never seen? They have no vectors ...
  - *Idea*: each word is a bag of characters ...
  - How can we use this?

fastText : n-grams

- A word w is represented as a bag of character n-grams
- We add special boundary words $<$ and $>$ to denote beginning and end of words
- Consider $w =$"where" and the $n = 3$ grams
  - These are "<wh", "whe", "her", "ere" and "re>"
  - Note we also add a special sequence "<where>" used to the actual word "where"
  - In practice, you can limit to grams between 3 and 6 in size

fastText : n-grams

- Given a word w, we let $\mathcal{G}_w$ the set of $n-$grams appearing in w
- We can, as before set

$$s(w_j, w_i) = \sum_{g \in \mathcal{G}_{w_j}} b_g^T a_{w_i}$$

- Now, we can share representations across words and learn more powerful representations, especially for rare words

fastText : Implementations Details

- Because there are so many $n-$grams, the authors hash each gram to a set of size 1 to K, where K is large
- Library is written in C++ with Python bindings
- A word like $w =$"where" can have a bunch of word vectors to learn
- For example, with 2, 3 grams you learn a vector for each of
  - "<wh", "whe", "her", "ere", "re>"
  - "<w", "wh", "he", "er", "re", "e>"
  - "<where>"
- $b_w = \frac{\sum_{g \in \mathcal{G}_w} b_g}{|\mathcal{G}_w|}$

fastText : Word Analogies

- How does fastText do on word analogies?
- Better on Syntactic, not on Semantic

		sg	cbow	sisg
CS	Semantic	25.7	27.6	27.5
	Syntactic	52.8	55.0	77.8
DE	Semantic	66.5	66.8	62.3
	Syntactic	44.5	45.0	56.4
EN	Semantic	78.5	78.2	77.8
	Syntactic	70.1	69.9	74.9
IT	Semantic	52.3	54.7	52.3
	Syntactic	51.5	51.8	62.7

Table 2: Accuracy of our model and baselines on word analogy tasks for Czech, German, English and Italian. We report results for semantic and syntactic analogies separately.

fastText : Text Similarity

- How does fastText rank words and how does this ranking compare to human ranking?

		sg	cbow	sisg-	sisg
AR	WS353	51	52	54	**55**
DE	GUR350	61	62	64	**70**
	GUR65	78	78	**81**	**81**
	ZG222	35	38	41	**44**
EN	RW	43	43	46	**47**
	WS353	72	**73**	71	71
ES	WS353	57	58	58	**59**
FR	RG65	70	69	**75**	**75**
RO	WS353	48	52	51	**54**
RU	HJ	59	60	60	**66**

Table 1: Correlation between human judgement and similarity scores on word similarity datasets. We train both our model and the `word2vec` baseline on normalized Wikipedia dumps. Evaluation datasets contain words that are not part of the training set, so we represent them using null vectors (`sisg-`). With our model, we also compute vectors for unseen words by summing the n-gram vectors (`sisg`).

fastText : Rare Words

- How does fastText do when it comes to rare words?

query	tiling	tech-rich	english-born	micromanaging	eateries	dendritic
`sisg`	tile flooring	tech-dominated tech-heavy	british-born polish-born	micromanage micromanaged	restaurants eaterie	dendrite dendrites
`sg`	bookcases built-ins	technology-heavy .ixic	most-capped ex-scotland	defang internalise	restaurants delis	epithelial p53

Table 7: Nearest neighbors of rare words using our representations and `skipgram`. These hand picked examples are for illustration.

fastText : Word Analogies

- How do you get the most important $n-$grams of a word?
- *Idea*: Compare b_w to $b_{w/g} = \sum_{g' \in \mathcal{G} - \{g\}} b_{g'}$
- ...The lowest ranking gram has the most information!
- Basically, you've pulled out the most information by not using the word vector for the gram in question

	word	n-grams		
DE	autofahrer	fahr	fahrer	auto
	freundeskreis	kreis	kreis>	<freun
	grundwort	wort	wort>	grund
	sprachschule	schul	hschul	sprach
	tageslicht	licht	gesl	tages
EN	anarchy	chy	<anar	narchy
	monarchy	monarc	chy	<monar
	kindness	ness>	ness	kind
	politeness	polite	ness>	eness>
	unlucky	<un	cky>	nlucky
	lifetime	life	<life	time
	starfish	fish	fish>	star
	submarine	marine	sub	marin
	transform	trans	<trans	form
FR	finirais	ais>	nir	fini
	finissent	ent>	finiss	<finis
	finissions	ions>	finiss	sions>

Table 6: Illustration of most important character n-grams for selected words in three languages. For each word, we show the n-grams that, when removed, result in the most different representation.

fastText : Supervised Classification

- fastText also has a supervised variant
- Essentially, text is turned into $n-$gram features and then a standard classifier is trained using (hierarchical) soft max objective
- This, along C++ is the "fast" part

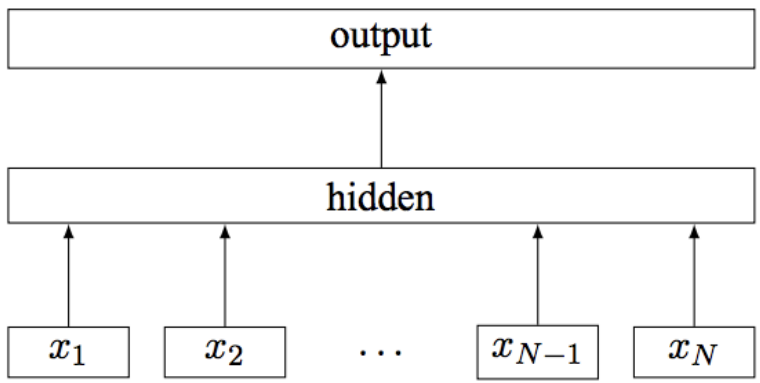


Figure 1: Model architecture of `fastText` for a sentence with N ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.

fastText : Supervised Classification

- Though not always the best, it is extremely fast and easy to use

	Zhang and LeCun (2015)		Conneau et al. (2016)			fastText
	small char-CNN	big char-CNN	depth=9	depth=17	depth=29	$h = 10$, bigram
AG	1h	3h	24m	37m	51m	1s
Sogou	-	-	25m	41m	56m	7s
DBpedia	2h	5h	27m	44m	1h	2s
Yelp P.	-	-	28m	43m	1h09	3s
Yelp F.	-	-	29m	45m	1h12	4s
Yah. A.	8h	1d	1h	1h33	2h	5s
Amz. F.	2d	5d	2h45	4h20	7h	9s
Amz. P.	2d	5d	2h45	4h25	7h	10s

Table 2: Training time for a single epoch on sentiment analysis datasets compared to char-CNN and VDCNN.

Model	AG	Sogou	DBP	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW (Zhang et al., 2015)	88.8	92.9	96.6	92.2	58.0	68.9	54.6	90.4
ngrams (Zhang et al., 2015)	92.0	97.1	98.6	95.6	56.3	68.5	54.3	92.0
ngrams TFIDF (Zhang et al., 2015)	92.4	97.2	98.7	95.4	54.8	68.5	52.4	91.5
char-CNN (Zhang and LeCun, 2015)	87.2	95.1	98.3	94.7	62.0	71.2	59.5	94.5
char-CRNN (Xiao and Cho, 2016)	91.4	95.2	98.6	94.5	61.8	71.7	59.2	94.1
VDCNN (Conneau et al., 2016)	91.3	96.8	98.7	95.7	64.7	73.4	63.0	95.7
fastText, $h = 10$	91.5	93.9	98.1	93.8	60.4	72.0	55.8	91.2
fastText, $h = 10$, bigram	92.5	96.8	98.6	95.7	63.9	72.3	60.2	94.6

Table 1: Test accuracy [%] on sentiment datasets. FastText has been run with the same parameters for all the datasets. It has 10 hidden units and we evaluate it with and without bigrams. For char-CNN, we show the best reported numbers without data augmentation.

Outline

- Glove
- fastText
- Skip Gram with Negative Sampling is Matrix Factorization

Skip-Gram with Negative Sampling is Matrix Factorization

- Recall that, for each center-context pair (w_o, w_c) objective of Skip-Gram with negative sampling is to maximize:

$$\log \sigma(b_{w_o}^{\mathsf{T}} a_{w_c}) + KE_{k \sim P(w)}[\log \sigma(-b_{w_k}^{\mathsf{T}} a_{w_c})]$$

- This is the log-probability of $p(w_o|w_c)$
- Note that we have made a simplification:

$$KE_{k \sim P(w)}[\log \sigma(-b_{w_k}^{\mathsf{T}} a_{w_c})] = \sum_{k=1}^{K} E_{k \sim P(w)}[\log \sigma(-b_{w_k}^{\mathsf{T}} a_{w_c})]$$

- The w_k is chosen randomly from the corpus and we postulate that on average we will never have seen w_k in the context of w_c

Skip Gram with Negative Sampling is Matrix Factorization

- If we take all words and pair them up, the objective of Skip-Gram becomes

$$\sum_{i=1}^{|V|}\sum_{j=1}^{|V|} N(w_i, w_j)[\log \sigma(b_{w_j}^{\mathsf{T}} a_{w_i}) + KE_{w_k \sim P(w)}[\log \sigma(-b_{w_k}^{\mathsf{T}} a_{w_i})]]$$

- $N(w_i, w_j)$ is the number of times w_j was seen in the context of center word w_i

Skip Gram with Negative Sampling is Matrix Factorization

- Notice that $N(w_i, w_j) = N(w_j, w_i)$
- $N(w_i) = \sum_{j=1}^{|V|} N(w_i, w_j)$ is the number of times w_i was seen as a center word
  - This is the same as the number of times w_i is seen as a context word
- $N = \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} N(w_i, w_j)$ is the number of center-context pairs we have
- Question: Can we simplify $KE_{w_k \sim P(w)}[\log \sigma(-b_{w_k}^{\mathsf{T}} a_{w_i})]$?

Skip Gram with Negative Sampling is Matrix Factorization

- We have

$$K * E_{w_k \sim P(w)}[\log \sigma(-b_{w_k}^{\mathsf{T}} a_{w_i})] = K \sum_{l=1}^{|V|} \frac{N(w_l)}{N} \log \sigma(-b_{w_l}^{\mathsf{T}} a_{w_i})$$

  - Here, we assume that a word can be picked from the corpus with the unigram distribution, $P(w) = N(w)/N$
  - In the original paper, they use $P(w) \propto N(w)^{3/4}$

Skip Gram with Negative Sampling is Matrix Factorization

- The objective of Skip-Gram can be written as

$$\sum_{i=1}^{V}\sum_{j=1}^{V} N(w_i, w_j)[\log \sigma(b_{w_j}^{\mathsf{T}} a_{w_i}) + KE_{w_k}[\log \sigma(-b_{w_k}^{\mathsf{T}} a_{w_i})]]$$

- Break this into two parts
- Positive part

$$\sum_{i=1}^{V}\sum_{j=1}^{V} N(w_i, w_j)[\log \sigma(b_{w_j}^{\mathsf{T}} a_{w_i})]$$

- Negative part

$$\sum_{i=1}^{V}\sum_{j=1}^{V} N(w_i, w_j)[K\sum_{l=1}^{V} \frac{N(w_l)}{N} \log \sigma(-b_{w_l}^{\mathsf{T}} a_{w_i})]$$

Skip Gram with Negative Sampling is Matrix Factorization

- Focus on the negative part
- Negative part

$$\sum_{i=1}^{V}\sum_{j=1}^{V} N(w_i, w_j)[K\sum_{l=1}^{V}\frac{N(w_l)}{N}\log\sigma(-b_{w_l}^{\mathsf{T}}a_{w_i})]$$

- Swap the order of indices we have that the above is

$$\sum_{i=1}^{V}\sum_{l=1}^{V}[\frac{KN(w_l)}{N}\log\sigma(-b_{w_l}^{\mathsf{T}}a_{w_i})\sum_{j=1}^{V} N(w_i, w_j)]$$

- This simplifies to

$$\sum_{i=1}^{V}\sum_{l=1}^{V}[\frac{KN(w_l)}{N}\log\sigma(-b_{w_l}^{\mathsf{T}}a_{w_i})N(w_i)]$$

Skip Gram with Negative Sampling is Matrix Factorization

- Notice l is a dummy variable so we can use j
- Negative part becomes

$$\sum_{i=1}^{V}\sum_{j=1}^{V}\frac{KN(w_i)N(w_j)}{N}\log\sigma(-b_{w_j}^{\mathsf{T}}a_{w_i})$$

- Bring the negative and positive parts together, you get

$$\sum_{i=1}^{V}\sum_{j=1}^{V}[N(w_i,w_j)\log\sigma(b_{w_j}^{\mathsf{T}}a_{w_i})+\frac{KN(w_i)N(w_j)}{N}\log\sigma(-b_{w_j}^{\mathsf{T}}a_{w_i})]$$

Skip Gram with Negative Sampling is Matrix Factorization

- Skip Gram with negative sampling is optimizing the objective below:

$$\sum_{i=1}^{V}\sum_{j=1}^{V}[N(w_i, w_j)\log\sigma(b_{w_j}^{\mathsf{T}}a_{w_i}) + \frac{KN(w_i)N(w_j)}{N}\log\sigma(-b_{w_j}^{\mathsf{T}}a_{w_i})]$$

- $b^{\mathsf{T}}a$ makes us think there is a matrix equation here, something like $BA^{\mathsf{T}} = M$ subject to the above objective
  - $A \in \mathbf{R}^{V\times d}$
  - $B \in \mathbf{R}^{V\times d}$
  - $M \in \mathbf{R}^{V\times V}$.
  - Question: What is M?

Skip Gram with Negative Sampling is Matrix Factorization

- Consider the gradient of the objective for a fixed product $b_{w_j}^{\mathsf{T}} a_{w_i}$
- $$L = \sum_{i=1}^{V} \sum_{j=1}^{V} [N(w_i, w_j) \log \sigma(b_{w_j}^{\mathsf{T}} a_{w_i}) + \frac{kN(w_i)N(w_j)}{N} \log \sigma(-b_{w_j}^{\mathsf{T}} a_{w_i})]$$
- For a fixed (i, j), take the derivative with respect to $b_{w_j}^{\mathsf{T}} a_{w_i}$

Skip Gram with Negative Sampling is Matrix Factorization

- We want the derivative of the below expression with respect to $x = b_{w_j}^{\mathsf{T}} a_{w_i}$, we want $\frac{\partial L}{\partial x}$

$$N(w_i, w_j) \log \sigma(x) + \frac{kN(w_i)N(w_j)}{N} \log \sigma(-x)$$

- We know: $(\log \sigma(x))' = \frac{\sigma'(x)}{\sigma(x)}$
- And also: $\sigma'(x) = \sigma(x)(1 - \sigma(x))$, $\sigma(-x) = 1 - \sigma(x)$
- we have that the above is

$$N(w_i, w_j)(1 - \sigma(b_{w_j}^{\mathsf{T}} a_{w_i})) - \frac{kN(w_i)N(w_j)}{N} \sigma(b_{w_j}^{\mathsf{T}} a_{w_i})$$

-

$$N(w_i, w_j)(1 - \sigma(b_{w_j}^{\mathsf{T}} a_{w_i})) - \frac{kN(w_i)N(w_j)}{N} \sigma(b_{w_j}^{\mathsf{T}} a_{w_i})$$

Skip Gram with Negative Sampling is Matrix Factorization

- We have that the above is

$$N(w_i, w_j) - (N(w_i, w_j) + \frac{kN(w_i)N(w_j)}{N})\sigma(b_{w_j}^{\mathsf{T}} a_{w_i})$$

- We have the (trick) $x = (x + y)\sigma(\log(x/y))$ so the above is
- We have that the above is

$$(N(w_i, w_j) + \frac{KN(w_i)N(w_j)}{N})\sigma(\log \frac{N(w_i, w_j)N}{KN(w_i)N(w_j)})$$

$$-(N(w_i, w_j) + \frac{KN(w_i)N(w_j)}{N})\sigma(b_{w_j}^{\mathsf{T}} a_{w_i})$$

- This is

$$[N(w_i, w_j) + \frac{KN(w_i)N(w_j)}{N}][\sigma(\log \frac{N(w_i, w_j)N}{KN(w_i)N(w_j)}) - \sigma(b_{w_j}^{\mathsf{T}} a_{w_i})]$$

Skip Gram with Negative Sampling is Matrix Factorization

- The derivative we seek is

$$(N(w_i, w_j) + \frac{KN(w_i)N(w_j)}{N})[\sigma(\log \frac{N(w_i, w_j)N}{KN(w_i)N(w_j)}) - \sigma(b_{w_j}^{\mathsf{T}} a_{w_i})]$$

- If this is zero for all (i, j), we have optimality
- Since the sigmoid is bijective, this is the same as

$$\log \frac{N(w_i, w_j)N}{KN(w_i)N(w_j)} = b_{w_j}^{\mathsf{T}} a_{w_i}$$

for any (i, j)

Skip Gram with Negative Sampling is Matrix Factorization

- We want, for any (i, j),

$$\log \frac{N(w_i, w_j)N}{KN(w_i)N(w_j)} = b_{w_j}^{\mathsf{T}} a_{w_i}$$

- So, Skip-Gram can be viewed as factoring M with $M_{ij} = \log \frac{N(w_i,w_j)N}{KN(w_i)N(w_j)}$, i.e. finding two matrices A and B so that $BA^{\mathsf{T}} = M$ but subject to the non-convex objective

$$\sum_{i=1}^{V}\sum_{j=1}^{V}[N(w_i, w_j)\log\sigma(b_{w_j}^{\mathsf{T}} a_{w_i}) + \frac{kN(w_i)N(w_j)}{N}\log\sigma(-b_{w_j}^{\mathsf{T}} a_{w_i})]$$

Skip Gram with Negative Sampling is Matrix Factorization

- $\log \frac{N(w_i,w_j)N}{N(w_i)N(w_j)}$ is famous
- Pointwise Mutual Information, $PMI_{ij} = \log \frac{N(w_i,w_j)N}{N(w_i)N(w_j)}$
  - Pointwise Mutual Information is the log of the log probability over the product of probabilities
  - $PMI_{ij} = \log \frac{P(w_i,w_j)}{P(w_i)P(w_j)}$

Skip Gram with Negative Sampling is Matrix Factorization

- $PMI_{ij} = \log \frac{P(w_i, w_j)}{P(w_i)P(w_j)}$
- We have $b_j^\intercal a_i = \log \frac{N(w_i, w_j)N}{KN(w_i)N(w_j)}$, but this reduces to
- $M_{ij} = PMI_{ij} - \log K$, so we are factoring a shifted mutual information matrix
- Note that $M_{ij} = -\infty$ implies $\sigma(M_{ij}) = 0$, so the loss function takes care of the case when M_{ij} is $-\infty$

Skip Gram with Negative Sampling is Matrix Factorization

- So, Skip-Gram can be viewed as factoring M with $M_{ij} = PMI_{ij} - logK$, i.e. finding two matrices A and B so that $BA^{\intercal} = M$ but subject to maximizing the non-convex objective

$$\sum_{i=1}^{V}\sum_{j=1}^{V}[N(w_i, w_j)\log\sigma(b_{w_j}^{\intercal}a_{w_i}) + \frac{KN(w_i)N(w_j)}{N}\log\sigma(-b_{w_j}^{\intercal}a_{w_i})]$$

Skip Gram with Negative Sampling is Matrix Factorization

- *We can optimize the Skip Gram problem in a different way, directly considering this optimization problem*
- When the Skip Gram model was built, this link was unknown
- Only recently was it flushed out (2019)

References

- Glove
- fastText
- fastText Python bindings
- Skip Gram with Negative Sampling is Matrix Factorization
- Word2Vec Explained
- Rohde's 2005 Paper on Word Vectors