Neural Bag-of-Ngrams

Bofang Li, Tao Liu, Zhe Zhao, Puwei Wang,* Xiaoyong Du
School of Information, Renmin University of China, Beijing, China
Key laboratory of Data Engineering and Knowledge Engineering, MOE, Beijing, China
{libofang, tliu, helloworld, wangpuwei, duyong}@ruc.edu.cn

Abstract

Bag-of-ngrams (BoN) models are commonly used for representing text. One of the main drawbacks of traditional BoN is the ignorance of n-gram's semantics. In this paper, we introduce the concept of Neural Bag-of-ngrams (Neural-BoN), which replaces sparse one-hot n-gram representation in traditional BoN with dense and rich-semantic n-gram representations. We first propose context guided n-gram representation by adding n-grams to word embeddings model. However, the context guided learning strategy of word embeddings is likely to miss some semantics for text-level tasks. Text guided n-gram representation and label guided n-gram representation are proposed to capture more semantics like topic or sentiment tendencies. Neural-BoN with the latter two n-gram representations achieve state-of-the-art results on 4 document-level classification datasets and 6 semantic relatedness categories. They are also on par with some sophisticated DNNs on 3 sentence-level classification datasets. Similar to traditional BoN, Neural-BoN is efficient, robust and easy to implement. We expect it to be a strong baseline and be used in more real-world applications.

Introduction

Text representation plays an important role in many natural language processing tasks. It aims at mapping varied-length texts (sentences, paragraphs, documents) into fixed-length vectors. The quality of text vectors will directly affect the downstream models' performance. Take text classification tasks for example, the way of representing texts is much more important than the choice of classifiers.

The most commonly used text representation model is bag-of-words (Joachims 1998), in which text is represented as the multiset of its belonging words. The grammar and word order are disregarded. Compared to bag-of-words, bag-of-ngrams considers not only word, but also consecutive words (n-gram). These models are often used as baselines in recent research and preferable in real-world applications due to their simplicity and robustness.

As shown in Figure 1, traditional bag-of-ngrams (BoN) can be regarded as the sum of n-gram vectors with one-hot representation. In one-hot representation, each n-gram is considered as an unique token which is different from each other absolutely. The semantics of n-grams is ignored under this condition. In this paper, we introduce Neural Bag-of-Ngrams (Neural-BoN) to overcome this drawback. It represents n-grams by dense, real-valued vectors instead of sparse vectors. N-grams with similar semantics are more likely to be similar in vector space. Text vector generated by summing neural n-gram vectors contains more semantics, which contributes more to the successive models.

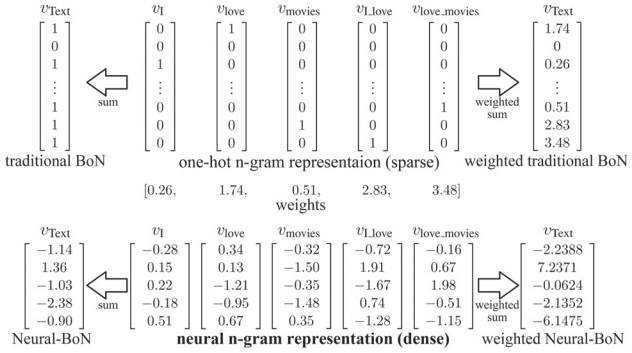

Figure 1: Example of traditional BoN and Neural-BoN for representing text "I love this movie".

In this paper, three types of neural n-gram representation (NR) are proposed: Context Guided N-gram Representation (CGNR), Text Guided N-gram Representation (TGNR) and Label Guided N-gram representation (LGNR) (Figure 2). As the name suggests, CGNR utilizes the n-gram co-occurrence information which lies in context. It is inspired by the recent success of word embeddings and is built on the basis of Skip-Gram (Mikolov et al. 2013). However, the context guided learning strategy of word embeddings and CGNR is likely to miss some semantics for text-level tasks. TGNR and LGNR are proposed to utilize the n-gram co-occurrence information which lies in text and texts' class labels respectively. They can capture more important information such as the topic or sentiment tendencies.

Neural-BoN inherits the advantages of both traditional BoN and neural word embeddings. It captures semantics with dense representation as neural word embeddings while it remains simple and robust as traditional BoN. Neural-

*Corresponding author.
Copyright © 2017, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

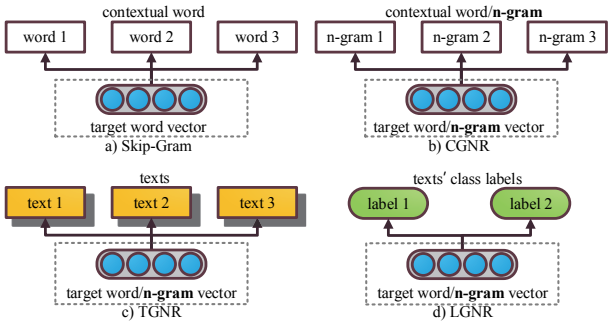

Figure 2: Skip-Gram and proposed n-gram representations.

BoN is also flexible, Weighting techniques like TF-IDF (Sparck Jones 1972) and Naive Bayes (Maron and Kuhns 1960) used in traditional BoN can be applied to Neural-BoN with little effort. Additional unlabeled corpora can also be used for training CGNR and TGNR, since they are unsupervised.

Related Work

Text Representation (TR) Models Text vectors can be generated based on word/n-gram vectors in a bottom-up fashion. Traditional BoW/BoN can be regarded as the sum of one-hot word/n-gram vectors. Recent researches (Mitchell and Lapata 2010; Socher et al. 2013; Hill, Cho, and Korhonen 2016) use sum/average of existing word embeddings as baselines for text representation. Word embeddings models learn word vectors by utilizing the word co-occurrence information which lies in "context". For example, **CBOW** and **Skip-Gram** (Mikolov et al. 2013) define the "context" of a target word as "the words surrounding it in a small window size". **Glove** (Pennington, Socher, and Manning 2014) uses the same definition and explicitly weights contextual words based on their position. **C-Phrase** (Pham et al. 2015) can be regarded as an improved version of CBOW which utilizes "syntactic context" indicated by syntactic parse tree. However, these word embeddings are not optimized for constructing texts. Compared to our model, they only consider words (uni-grams) and their co-occurrence in context. More powerful vectors can be learned by introducing n-grams and other types of co-occurrence information which lies in text and texts' class.

Instead of summing word vectors, another line of methods learn text representations directly. In **Paragraph Vector** (PV) (Le and Mikolov 2014), paragraph (text) vector is learned to be useful for predicting its belonging target words. Our text guided n-gram representation (TGNR) can be regarded as a reverse and more general (n-gram) version of PV. It learns n-gram vector by predicting which text it belongs. PV and the bottom-up models don't consider word order or only consider word order in short range context. More complex Deep Neural Networks (DNN) can be used for modeling word order in long range context. For example, **Recursive Auto-encoder** (Socher et al. 2011) assigns a vector for each node in a sentence's syntactic parse tree and represents the sentence with the root node's vector. Each node's vector encodes the information of its subtree and is learned by reconstructing its child nodes. In **Skip-Thought** (Kiros et al.), sentence vector is generated by Recurrent Neural Network (RNN) and is learned to be useful for predicting its surrounding sentences' representation. In (Hill, Cho, and Korhonen 2016), two modifications of Skip-Thought are proposed for fast learning and more general usage: **Fast Sentence** (Fast) simplifies Skip-Thought by predicting which words in the surrounding sentence instead of predicting surrounding sentence itself. **Sequential Denoising Autoencoders** (SAE) learns sentence representation by predicting its corrupted version, thus no surrounding sentences are needed. Compared to Neural-BoN, these models are limited to sentence representation and are time consuming. Furthermore, since their architectures are complex, the noises caused by large parameters can hurt the models' performance.

Implicit Text Representation (ITR) Models There are also a lot of DNNs which generate text representations implicitly, such as Recurrent Neural Network (RNN) (Dai and Le 2015), Recursive Neural Network (RecNN) (Socher et al. 2013), Convolutional Neural Network (CNN) (Kim 2014), and their combinations (Cho et al. 2014; Lai et al. 2015) and variations (Johnson and Zhang 2015; Zhang, Zhao, and LeCun 2015; Tang, Qin, and Liu 2015). These ITR models focus only on text classification task and often achieved state-of-the-art results. After these models are trained, the layer just before the output layer or any fixed-length layer can be regarded as the input text's representation. However, these text representations can only capture information needed for a specific task and are not suitable for general usage, which fall out of the scope of this paper. Nonetheless, we compare our models with ITR models on text classification task. The results suggested our more general text representations can be on par with these ITR models.

Model

The key part of Neural-BoN is to learn meaningful word and n-gram vectors for the construction of text vectors. In this section, we first propose three types of n-gram vector learning models. We then show how to construct text vectors using Neural-BoN and weighting techniques.

Context Guided N-gram Representation

N-gram is an important feature for understanding text. For example in Table 1, bi-gram "not good" in $Text_1$ expresses the negative sentiment and is more important than words "good" and "not". Unlike word embeddings models which consider only words, Context Guided N-gram Representation (CGNR) learns the vector of n-grams such as "not good" to capture its negative sentiment directly.

CGNR is motivated by word embedding models, especially skip-Gram (Mikolov et al. 2013). Skip-Gram is efficient to train, scales well to huge corpora, and is very robust as shown in (Levy and Goldberg 2014; Levy, Goldberg, and Dagan 2015). CGNR and Skip-Gram have the same learning

Table 1: Illustration of some texts and their sentiments.

ID	Sentiment	Text
$Text_1$	negative	This film is **not good**.
$Text_2$	positive	This film is **good**.
$Text_3$	negative	This film is **bad**.
$Text_4$	positive	This film is **good**, I give **7/10** to it.
$Text_5$	positive	Patrick Swayze's acting is **perfect**.

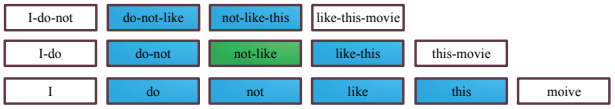

Figure 3: Contextual n-grams set (denoted by blue boxes) for target bi-gram "not like" (denoted by green box). $win = 1$. $m = 3$ (tri-gram model).

strategy: n-gram vectors are learned to be useful for predicting its context. Actually, CGNR can be regarded as a more general version of Skip-Gram which considers contextual n-grams instead of words. To be more precise, the objective function of CGNR can be formalized as:

$$\sum_i \sum_j \sum_q \log p\left(c_q^{g_{n,i,j}} | v_{g_{n,i,j}}\right) \tag{1}$$

where $g_{n,i,j}$ denotes the j^{th} n-gram from i^{th} text, $c_q^{g_{n,i,j}}$ denotes its q^{th} contextual n-gram. v_g denotes the vector of n-gram g. The contextual n-gram set of the target n-gram $g_{n,i,j}$ can be defined as:

$$\begin{aligned} c^{g_{n,i,j}} = \{ & g_{nc,i,j+t} \text{ where } 1 \leq nc \leq m \\ & \text{and } -win \leq t \leq win + n - nc \} \end{aligned} \tag{2}$$

where win is the contextual window size and m is the maximum size of gram. An example of contextual n-gram is shown in Figure 3.

The prediction in Equation 1 is theoretically defined as softmax:

$$p\left(c_q^{g_{n,i,j}} | v_{g_{n,i,j}}\right) = \exp(y_{c_q^{g_{n,i,j}}})/Z \tag{3}$$

where $y_{c_q^{g_{n,i,j}}}$ is the un-normalized probability for q^{th} contextual n-gram of $g_{n,i,j}$ given input n-gram vector $v_{g_{n,i,j}}$. Z denotes the normalization factor. The vector y is computed as:

$$y = W v_{g_{n,i,j}} + b \tag{4}$$

where W and b are the softmax parameters. In this way, the vectors of n-grams with similar contexts are learned by similar prediction, thus are clustered together in vector space.

However, n-grams with similar contexts may not have the same semantics. This learning strategy of word embeddings models and CGNR is insufficient and even problematical for some text-level tasks. For example, uni-grams "good" and "bad" are both transitive verbs and their context is similar in most corpora as illustrated in Table 1. Their vectors learned by CGNR are actually nearest neighbors according to our experimental results in Table 2. This result is reasonable for some word-level tasks like POS tagging. But for text-level tasks like text classification and semantic relatedness, "good" and "bad" express totally different semantics and should be far away from each other in vector space. This motivates us to propose another two n-gram representation learning models.

Table 2: Experimental results of n-gram representations (NR), which are trained on IMDB datasets. Superscript ^{m} indicates movie name. Superscript ^{a} indicates actor name.

N-gram	NR	Nearest Neighbours
good	CGNR	decent, damn, bad, old-fashioned, passable, so-so very good, good movie, not good, actually pretty
	TGNR	decent, pretty, well-done, 7/10, passable, appealing good movie, good acting, very good, good director
	LGNR	sobieski^a, katrina^a, ponyo^m, perfect, gulliver^m lonesome dove^m, patrick swayze^a, batman returns^m
not good	CGNR	so-so, bad, appalling, terrible, good, acceptable not bad, not great, particularly good, plain bad
	TGNR	fault, bad, okay, terrible, horrible, bearable, 3/10 not great, not well, bad ones, no good, not enough
	LGNR	carly^a, revolting, herzog^a, critters^m, nauseating robert young^a, worst movie, steven seagal^a, 2 stars

Text Guided N-gram Representation

Consider uni-gram "good" and "7/10" (a relatively high review score) in Table 1, these two uni-grams have totally different parts of speech and tend to appear in different contexts. However, they both express positive attitude and this information is crucial for text-level tasks, especially sentiment classification. This observation suggests n-grams that appear in the same text tend to have similar semantics. TGNR captures this information by clustering n-grams which appear in the same text together in vector space. To be more precise, n-gram vector is learned to be useful for predicting which text it belongs to:

$$\sum_i \sum_j \log p\left(t_i | v_{g_{n,i,j}}\right) \tag{5}$$

where t_i denotes the i^{th} text.

TGNR works especially well for long texts (documents). For example, a long negative movie review is likely to contain many n-grams like "terrible", "waste of time" and "no good". In TGNR, these negative n-grams are clustered together in vector space. On the other hand, a short movie review may contain only one sentiment n-gram. It is hard for TGNR to cluster this n-gram with any other sentiment n-grams.

Label Guided N-gram Representation

In Table 1, uni-gram "good" and "perfect" are similar since they express the same positive sentiment. Both CGNR and TGNR can not capture this similarity since these uni-grams appear in different contexts and texts. Actually no model can capture this similarity without more texts or prior knowledge.

In the case of text classification, each text in the training set is assigned to a class. Therefore, uni-gram "good" and "perfect" can be learned to be similar, since they appear in texts with the same positive sentiment label.

In traditional BoN, Naive Bayes weighting (NB) (Wang and Manning 2012) is used to capture this labeling information. NB directly weights each n-gram based on its frequencies in text classes. LGNR can be regarded as a dense version of NB, which implicitly capture weights by predicting texts' class label:

$$\sum_i \sum_j \log p\left(l_{t_i} | v_{g_{n,i,j}}\right) \tag{6}$$

where l_t denotes the class label of text t.

In contrast to TGNR, LGNR works especially well for short texts (sentence) and small datasets. Compared to long texts, labels for short texts are more specific and accurate. N-gram vectors learned by these labeled short texts contain less noises.

Both TGNR and LGNR can overcome the problem existed in CGNR. In these two models, uni-gram "good" and "bad" are learned to be far away from each other because they appear in different texts and texts' labels. Note that unlike CGNR and TGNR, LGNR is supervised and needs labeled text.

Weighted Neural Bag-of-Ngrams

After n-gram representations are learned, the simplest way of constructing a text vector is summing its belonging n-grams vectors. However, different n-grams have different impact on a text. Since traditional bag-of-ngram models can also be regarded as the sum of n-gram vectors, weighting techniques used in them can be applied to Neural-BoN directly as shown in Figure 1. In this paper, TF-IDF (Sparck Jones 1972) and Naive Bayes (NB) weighting (Maron and Kuhns 1960) is considered.

Computational Complexity

As shown in Equation 5 and Equation 6, training TGNR and LGNR for one epoch only needs to scan the training corpus C once. With Negative Sampling technique, the probability p is calculated by the inner product of n-gram vector and negative vector for $K+1$ times, where K is negative sampling size. The computational complexity of training TGNR/LGNR for one epoch is $O(|C|Kdm)$, where $|C|$ is corpus size, d is vector dimension and m is the maximum size of gram. As for CGNR, the training is $2w$ times slower than TGNR and LGNR, since a window size w needs to be iterated (Equation 1 and Equation 2).

In contrast, since almost every DNN needs matrix multiplication, the computational complexity of training them can be estimated as $O(|C|td^2)$, where t is the matrix multiplication times. Since K, m, w and t are relatively small compared to d and $|C|$, Neural-BoN is roughly d times faster than DNNs in theory.

Empirically, matrix multiplications in DNNs can benefit from GPU, especially for CNNs. However, Neural-BoN on a multi-core CPU is still much faster than DNNs. Table 3 lists the approximate training time of models for a single epoch on one million words.

Table 3: Approximate training time of models for a single epoch on one million words. CPU: Intel Xeon E5-2670 (32-core). GPU: NVIDIA Tesla K40.

model	device	training time
Neural-BoN (bi-gram)	CPU	0.6h
CNN	GPU	16h
Character-level CNN	GPU	109h
SDAE	GPU	54h
Skip-Thought	GPU	255h

Experiments

In order to better understand the learned text representations, we perform qualitative evaluation on IMDB dataset (Table 2), and quantitative evaluation on text classification task (7 datasets) and semantic relatedness task (2 datasets with 7 categories).

Training Details

In practice, the vocabulary size and the number of texts can be large, computing the softmax function in CGNR and TGNR is time consuming. Negative Sampling (Mikolov et al. 2013) is used for speeding up. N-gram vectors are first randomly initialized and then trained using stochastic gradient descent where the gradient is obtained via backpropagation (Williams and Hinton 1986).

For text classification task, hyper-parameters are tuned on 20% of the training data from IMDB dataset (Maas et al. 2011). For semantic relatedness task, hyper-parameters are tuned on the development data from SICK dataset (Marelli et al. 2014). Optimal hyper-parameters are actually identical: the vector dimension is 500, the learning rate is fixed to 0.25, the negative sampling size is 5, and models are trained for 10 iteration. Unlike most other neural models, Neural-BoN needs fewer hyper-parameters [1] and thus requires less tuning, which makes it easier to be applied to other tasks and real-world applications.

Text Classification

Text classification task aims at assigning a text with a predefined category. We evaluate our models on 3 sentence-level and 4 document-level datasets. More detailed statistics are shown in Table 4. For this task, text vectors are first normalized and considered as features for the classifier. We use Logistic Regression Classifier (Fan et al. 2008) in all of our experiments. Accuracy is used as evaluation metrics.

[1] Neural-BoN doesn't need hyper-parameters like number of layers, hidden layer size, mini-batch size, truncated BPTT length (for RNN), number of feature maps and pooling type (for CNN). Note that DNNs need to tune the size of each hidden layer, while Neural-BoN only needs to tune word vector's dimension.

Table 4: Datasets statistics. #Texts: the number of training and test texts. CV: the number of cross-validation splits, where N denotes default train/test split provided in dataset. #Tokens: the number of tokens. $|V|$: vocabulary size. #N-gram/T: the average number of n-grams contained in per text.

Item		MR	CR	Subj.	AthR	XGraph	RT-2k	IMDB	STS	SICK
domain		sentiment	customer review	subjective review	news	news	sentiment	sentiment	-	-
CV		10	10	10	N	N	10	N	-	-
#Texts		10,662	10,624	10,000	1,427	1,953	2,000	50,000	9,000	18,854
Gram	Item	MR	CR	Subj.	AthR	XGraph	RT-2k	IMDB	STS	SICK
Uni	#Tokens	224K	76K	241K	458K	458K	1493K	13055K	80K	181K
	$\|V\|$	21K	5.7K	24K	22K	32K	51K	171K	14K	2K
	#N-gram/T	21	20	24	321	234	746	261	9	10
Bi	#Tokens	437K	148K	471K	950K	980K	2983K	26059K	159K	362K
	$\|V\|$	133K	40K	148K	185K	206K	519K	2351K	48K	12K
	#N-gram/T	41	39	47	666	501	1491	521	18	19
Tri	#Tokens	640K	216K	692K	1370K	1368K	4472K	39014K	239K	544K
	$\|V\|$	308K	96K	340K	478K	490K	1560K	8894K	86K	31K
	#N-gram/T	60	57	69	960	700	2236	780	27	29

Table 5: Effect of different n-gram representations (NR) for text classification task. Best results overall are Underlined while best results in group are **Bold**.

Gram	NR	sentence-level			document-level			
					small vocabulary		large vocabulary	
		MR	CR	Subj	AthR	XGraph	RT-2k	IMDB
Uni	CGNR	69.10	76.42	90.73	74.54	84.02	80.3	84.06
	TGNR	64.00	73.09	87.65	83.03	86.99	**88.1**	**90.24**
	LGNR	**77.92**	**79.95**	**92.12**	**86.96**	**89.86**	83.2	85.06
+Bi	CGNR	71.76	77.03	91.98	76.72	86.06	83	84.63
	TGNR	69.79	77.19	88.32	84.01	87.81	**88.75**	**91.64**
	LGNR	**78.89**	**81.69**	**93.31**	**89.9**	**92.42**	86.5	87.15
+Bi+Tri	CGNR	69.39	75.79	90.52	74.47	84.42	83.1	85.35
	TGNR	63.25	73.96	88.23	83.87	87.39	**88.8**	**91.83**
	LGNR	**78.22**	**81.46**	**92.80**	**89.2**	**91.29**	85.6	87.48

Table 6: Comparison with other models on IMDB datasets. Top group: TR models. Bottom group: ITR models.

Model	IMDB
Maas (Maas et al. 2011)	87.99
PV (Mesnil et al. 2014)	88.73
NBSVM (Wang and Manning 2012)	91.22
best one-hot+NB	91.87
our model (TGNR)	**93.51**
RNN-LM (Mikolov 2012)	86.60
DAN (Iyyer et al. 2015)	89.4
DCNN (Iyyer et al. 2015)	89.4
SA-LSTM (Dai and Le 2015)	92.76
CNN+U3 (Johnson and Zhang 2015)	**93.49**

Default Scenario We first consider the default scenario where n-gram vectors are learned solely on the given classification dataset. No additional unlabeled corpora or weighting techniques are used. The following observations can be made from the results in Table 5:

- Compared to word (uni-gram) vectors, adding bi-gram consistently improves the performance for all n-gram representations across all datasets. However, adding tri-gram only slightly improve the performance on large datasets like RT-2k and IMDB. In small datasets, since most tri-grams appear only a few times, they are likely to bring noises to the model.
- TGNR outperforms CGNR on all document-level datasets. Compared to sentence-level datasets, texts in document-level datasets contain more n-grams. N-gram vectors learned on these datasets are more likely to capture useful information.
- LGNR performs best on all datasets except RT-2k and IMDB. It directly captures texts' class information, which is most useful for text classification. However, for datasets with large vocabulary (RT-2k and IMDB), it's hard for texts' class to distinguish all these n-grams.

We use tri-gram for large datasets (RT-2k and IMDB), and bi-gram for others in the following experiments.

Model's Improvements Unlabeled corpora often contain more information than single dataset and can potentially improve the performance. On three movie review datasets (MR, RT-2k and IMDB), Neural-BoN is trained along with unlabeled corpus from IMDB dataset, the same way as (Maas et al. 2011; Mesnil et al. 2014; Le and Mikolov 2014). This idea is also commonly used in neural networks like RNN (Zhao, Lu, and Poupart 2015) and CNN (Kim 2014; Iyyer et al. 2015; Johnson and Zhang 2015), where the input word vectors are pre-trained on large corpora.

We have also tried other non-sentiment corpora such as STATMT NEWS and Wikipedia corpora. However, they can't improve the accuracy of text classification. We conclude that only adding domain-related corpus can improve the models performance. It can also be confirmed in Table6, where IMDB corpus can only improve the performance on sentiment-related datasets, but not on others.

We choose Naive Bayes (NB) weighting in this experiment since it consistently outperforms TF-IDF on text classification task.

We also combine Neural-BoN's representations with traditional BoN's one-hot representation. This ensemble is

Table 7: Models' improvements and comparison with previous state-of-the-art results (SOA). SOAs are grouped as text representation (TR) models and implicit text representation (ITR) models. LGNR can't make use of additional unlabeled corpus (+corpus) since it requires labeled text. It also can't benefit from weights since it already contains label information.

N-gram Representation (NR)		MR	CR	Subj	AthR	XGraph	RT-2k	IMDB
CGNR	-	71.76	77.03	91.98	76.72	86.06	83.1	85.35
	+corpus	76.03(+4.27)	-	-	-	-	86(+2.9)	86(+0.65)
	+NB	77.6(+1.57)	78.68(+1.65)	92.24(+0.26)	78.26(+1.54)	87.7(+1.63)	86.5(+0.5)	88.95(+2.95)
	+one-hot	79.66(+2.02)	81.8(+3.12)	92.86(+0.62)	85.69(+7.43)	91.59(+3.89)	89.4(+2.9)	91.60(+3.65)
TGNR	-	69.79	77.19	88.32	84.01	87.81	88.8	91.83
	+corpus	79.25(+9.46)	-	-	-	-	90.9(+2.1)	92.09(+0.26)
	+NB	80.15(+0.9)	77.72(+0.53)	92.11(+0.28)	84.71(+0.7)	88.72(+0.91)	91.35(+0.45)	92.68(+0.59)
	+one-hot	**81.06**(+1.91)	81.93(+4.21)	92.79(+0.68)	88.35(+3.64)	90.88(+2.16)	**91.95**(+0.6)	**93.51**(+0.83)
LGNR	-	78.89	81.69	93.31	89.9	92.42	86.5	87.48
	+one-hot	79.55(+0.66)	**82.41**(+0.72)	93.41(+0.1)	**90.6**(+0.7)	**92.82**(+0.6)	88.7(+2.2)	91.37(+3.89)
TR-SOA		79.4 (NBSVM)	81.8 (NBSVM)	**93.6** (Skip-Thought)	87.7 (NBSVM)	90.7 (NBSVM)	89.45 (NBSVM)	91.22 (NBSVM)
ITR-SOA		**83.1** (AdaSent)	**86.3** (AdaSent)	**95.5** (AdaSent)	85.1 (MNB)	91.2 (MNB)	90.2 (Appr.T)	93.49 (CNN)

Table 8: Comparison with other models on sentence-level datasets. Top group: TR models. Bottom group: ITR models.

Model	MR	CR	Subj
CPHRASE (Pham et al. 2015)	75.7	78.8	91.1
PV (Le and Mikolov 2014)	74.8	78.1	90.5
Skip-Thought (Kiros et al.)	76.5	80.1	**93.6**
best one-hot+NB	78.43	81.16	92.27
NBSVM (Wang and Manning 2012)	79.4	81.8	93.18
our model	**81.06**	**82.41**	93.41
GrConv (Cho et al. 2014)	76.3	81.3	89.5
RNN (Zhao, Lu, and Poupart 2015)	77.2	82.3	93.7
CNN (Kim 2014)	81.5	85.0	93.4
BRNN (Zhao, Lu, and Poupart 2015)	82.3	82.6	94.2
AdaSent (Zhao, Lu, and Poupart 2015)	**83.1**	**86.3**	**95.5**

commonly used in previous models for text classification (Maas et al. 2011; Dahl, Adams, and Larochelle 2012; Mesnil et al. 2014; Johnson and Zhang 2015).

The results of above improvements are shown in Table 7.

Comparison Table 6 and Table 8 show more detailed comparison of models. On sentence-level datasets, ITR models are still dominant. It's reasonable since ITR models focus on classification and are highly optimized for the specific task. Our models, along with other TR models, focus on text representation and are trained for general usage. Still, our model outperforms or is on par with ITR models like CNN and RNN, while needs much less time to train. Neural-BoN also outperforms previous text representation (TR) models on all datasets except Subj.

Document-level datasets are previously dominated by SVM with different features. Most ITR models are designed to capture word order information in long range contexts. This information is less crucial than that in sentence-level tasks. Therefore, their complex architectures become burdensome: introducing noises while providing less useful information. Neural-BoN achieves new state-of-the-art results on these datasets, as shown in Table 7.

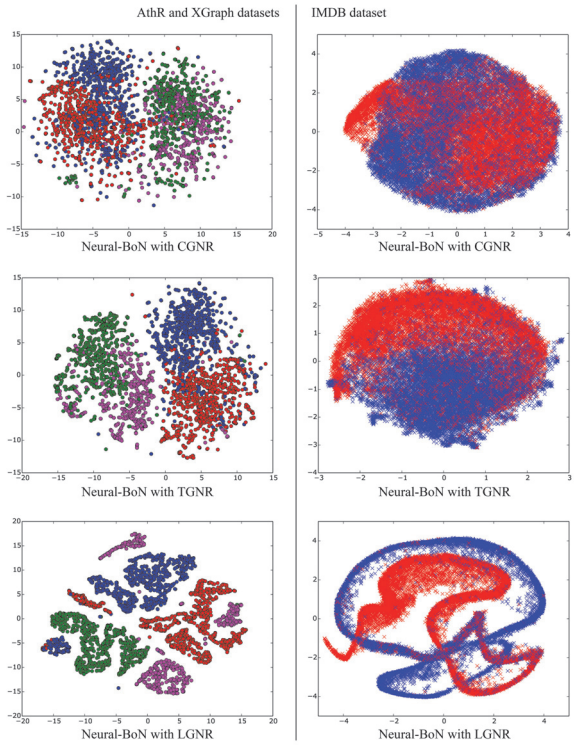

Figure 4: Visualization of text vectors. Different colors represent different text classes. Models are trained using bigram without any additional corpus or weighting techniques.

Visualization We also visualize text vectors learned by Neural-BoN. As shown in Figure 4, all of our proposed models have the ability of clustering text in the same class together. It is a very interesting property especially for CGNR

Table 9: Experimental results (Spearman/Pearson correlations) on semantic relatedness datasets. Ordered corpus requires the target text to be associated with contextual texts. Best results overall are Underlined while best results in group are **Bold**.

Corpus requirement	Gram	NR	STS						SICK
			News	Forum	WordNet	Twitter	Images	Headlines	
none	Uni	CGNR	0.62/0.64	0.36/0.37	0.71/0.67	0.67/0.73	0.67/0.69	0.58/0.60	0.59/0.64
		TGNR	**0.65/0.69**	**0.38/0.39**	**0.75/0.72**	**0.68/0.73**	**0.75/0.79**	**0.59/0.61**	**0.59/0.73**
	+Bi	CGNR	0.56/0.59	0.39/0.40	0.71/0.68	0.67/0.70	0.61/0.62	0.53/0.54	0.60/0.64
		TGNR	**0.61/0.63**	**0.44/0.45**	**0.76/0.74**	**0.69/0.71**	**0.73/0.76**	**0.57/0.59**	**0.61/0.74**
	SAE		0.17/0.16	0.12/0.12	0.30/0.23	0.28/0.22	0.49/0.46	0.13/0.11	0.32/0.31
	SAE+embs.		**0.52/0.54**	0.22/0.23	**0.60/0.55**	**0.60/0.60**	**0.64/0.64**	0.41/0.41	**0.47/0.49**
	SDAE		0.07/0.04	0.11/0.13	0.33/0.24	0.44/0.42	0.44/0.38	0.36/0.36	0.46/0.46
	SDAE+embs.		0.51/**0.54**	0.29/0.29	0.56/0.50	0.57/0.58	0.59/0.59	0.43/0.44	0.46/0.46
	PV-DBOW		0.31/0.34	0.32/0.32	0.53/0.50	0.43/0.46	0.46/0.44	0.39/0.41	0.42/0.46
	PV-DM		0.42/0.46	**0.33/0.34**	0.51/0.48	0.54/0.57	0.32/0.30	**0.46/0.47**	0.44/0.46
	one-hot+TF-IDF		0.48/0.48	0.40/0.38	0.60/0.59	0.63/0.65	0.72/0.74	0.49/0.49	0.52/0.58
ordered	SkipThought		0.44/0.45	0.14/0.15	0.39/0.34	0.42/0.43	0.55/0.60	0.43/0.44	0.57/0.60
	FastSent		**0.58/0.59**	**0.41**/0.36	**0.74/0.70**	0.63/0.66	**0.74/0.78**	0.57/0.59	**0.61/0.72**
	FastSent+AE		0.56/**0.59**	**0.41/0.40**	0.69/0.64	**0.70/0.74**	0.63/0.65	**0.58/0.60**	0.60/0.65

and TGNR, since they are learned without text class information. From clustering perspective alone, TGNR works better than CGNR, and LGNR works best. However, text vectors in LGNR is over clustered. It's hard for the successive classifier to remedy cluster error from LGNR. LGNR may not perform as good as it seems and the quantitative evaluation results (e.g. TGNR outperforms LGNR on IMDB dataset) also confirm this.

Semantic Relatedness

Semantic relatedness task aims at producing a semantic relatedness score of a text pair, which is compared with the human label. In contrast to text classification task which evaluates the quality of text representations by the performance of successive classifier, semantic relatedness task directly evaluates the quality of text representations by taking their cosine distance as the relatedness score.

The SICK (Marelli et al. 2014) and STS (Agirre et al. 2014) datasets are used for this task, the same as (Hill, Cho, and Korhonen 2016). Similar to previous researches, Toronto Books Corpus [2] is used as training data. Unlike text classification task, semantic relatedness task provides no texts' labels. LGNR model is unsuitable for this task and NB weighting technique is not used for models' improvements. The lack of labels also excludes implicit text representation (ITR) models. In order to make fair comparison, models which use structured resources (e.g. dictionary) are not considered in this experiment.

Several observations can be drawn from Table 9:

- The importance of word order is unclear on semantic relatedness task. Adding bi-gram improves the performance of Neural-BoN on SICK dataset, Forum and WordNet categories of STS dataset and . Adding tri-grams hurts the performance slightly on all categories. The competitive results of FAST model also support this claim, since it ignores word order.
- TGNR outperforms CGNR on all categories. This further supports our claim that n-gram vectors learned by considering n-grams co-occurrence in context are insufficient.
- Our model achieves state-of-the-art results on all categories except Twitter. Twitter category contains many rare n-grams (not in training corpus or appear very few times). This limits Neural-BoN to fully learn their n-gram vectors. In contrast to Neural-BoN, traditional BoN (one-hot+TFIDF) can make use of each n-gram and obtains good performance on this category.

Conclusion and Future Work

In this paper, we introduce the concept of Neural-BoN, which learns text vector by summing its belonging neural n-gram vectors (with weights). [3] Compared to its unigram version, adding bi-grams improves the performance of Neural-BoN on most datasets, while further adding trigrams only improves its performance on large datasets. We propose three types of n-gram representations and demonstrate their effectiveness on text classification task and semantic relatedness task: (1) Context Guided N-gram Representation (CGNR) uses the same idea as traditional word embeddings and is problematical for text-level tasks. (2) Text Guided N-gram Representation (TGNR) performs consistently well and especially suitable for document-level dataset with large vocabulary. (3) Label Guided N-gram Representation (LGNR) is more suitable for small datasets and implicitly contains NB weighting.

Our model achieves the new state-of-the-art results on 4 document-level classification datasets and 6 semantic relatedness categories. Inspired by these results, in future work, we will consider neural text representations beyond bag-of-ngrams. For example, the weighted sum of TGNR/LGNR can be replaced by composition functions based on syntactic parse tree or document structure.

[2] http://www.cs.toronto.edu/~mbweb/

[3] The source code of Neural-BoN is published at https://github.com/libofang/Neural-BoN.

Acknowledgments

This work is supported by National Natural Science Foundation of China (Grant No. 61472428 and No. 71271211), the Fundamental Research Funds for the Central Universities, the Research Funds of Renmin University of China No. 14XNLQ06. This work is partially supported by ECNU-RUC-InfoSys Joint Data Science Lab and a gift from Tencent.

References

Agirre, E.; Banea, C.; Cardie, C.; Cer, D.; Diab, M.; Gonzalez-Agirre, A.; Guo, W.; Mihalcea, R.; Rigau, G.; and Wiebe, J. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *SemEval*, 81–91.

Cho, K.; van Merrienboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *EMNLP*, 103–111.

Dahl, G. E.; Adams, R. P.; and Larochelle, H. 2012. Training restricted boltzmann machines on word observations. In *ICML*.

Dai, A. M., and Le, Q. V. 2015. Semi-supervised sequence learning. In *NIPS*, 3079–3087.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research* 9:1871–1874.

Hill, F.; Cho, K.; and Korhonen, A. 2016. Learning distributed representations of sentences from unlabelled data. In *HLT-NAACL*, 1367–1377.

Iyyer, M.; Manjunatha, V.; Boyd-Graber, J. L.; and III, H. D. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL*, 1681–1691.

Joachims, T. 1998. Text categorization with suport vector machines: Learning with many relevant features. In *ECML*, 137–142.

Johnson, R., and Zhang, T. 2015. Effective use of word order for text categorization with convolutional neural networks. In *NAACL*, 103–112.

Kim, Y. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, 1746–1751.

Kiros, R.; Zhu, Y.; Salakhutdinov, R.; Zemel, R. S.; Torralba, A.; Urtasun, R.; and Fidler, S. Skip-thought vectors. In *NIPS*.

Lai, S.; Xu, L.; Liu, K.; and Zhao, J. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, 2267–2273.

Le, Q. V., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *ICML*, 1188–1196.

Levy, O., and Goldberg, Y. 2014. Dependency-based word embeddings. In *ACL*, 302–308.

Levy, O.; Goldberg, Y.; and Dagan, I. 2015. Improving distributional similarity with lessons learned from word embeddings. *TACL* 3:211–225.

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *ACL*, 142–150.

Marelli, M.; Menini, S.; Baroni, M.; Bentivogli, L.; Bernardi, R.; and Zamparelli, R. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*, 216–223.

Maron, M. E., and Kuhns, J. L. 1960. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM (JACM)* 7(3):216–244.

Mesnil, G.; Ranzato, M.; Mikolov, T.; and Bengio, Y. 2014. Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. In *ICLR workshop*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, 3111–3119.

Mikolov, T. 2012. Statistical language models based on neural networks. *PhD thesis*.

Mitchell, J., and Lapata, M. 2010. Composition in distributional models of semantics. *Cognitive Science* 34:1388–1429.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP*, 1532–1543.

Pham, N. T.; Kruszewski, G.; Lazaridou, A.; and Baroni, M. 2015. Jointly optimizing word representations for lexical and sentential tasks with the c-phrase model. In *ACL*, 971–981.

Socher, R.; Huang, E. H.; Pennin, J.; Manning, C. D.; and Ng, A. Y. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*, 801–809.

Socher, R.; Perelygin, A.; Wu, J. Y.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, volume 1631, 1642. Citeseer.

Sparck Jones, K. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* 28(1):11–21.

Tang, D.; Qin, B.; and Liu, T. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, 1422–1432.

Wang, S. I., and Manning, C. D. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*, 90–94.

Williams, D. R. G. H. R., and Hinton, G. 1986. Learning representations by back-propagating errors. *Nature* 323–533.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *NIPS*, 649–657.

Zhao, H.; Lu, Z.; and Poupart, P. 2015. Self-adaptive hierarchical sentence model. In *IJCAI*, 4069–4076.