NLP 中的 Sentence Representation

Strong baseline		Bag-of-Words→(SIF 加强版)
State-of-the-art	Unsupervised	Skip-Thoughts
		Quick-Thoughts
	Supervised	InferSent
	Multi-task learning	MILA/MSR'S General purpose Sent
		Google's Universal Sentence Enc

对于 NLP 方面的工作来说，词嵌入可以说是最重要的基石，随着研究的深入，越来越多的人关心如何利用词嵌入表达好一个句子或是一段话，从 word embedding 上升到 sentence embedding。

下面总结了一下 sentence embedding 的生成方法。

一、组合抽象

常见的就是从 word embedding 组合抽象得到 sentence embedding，如：

Bag of Words

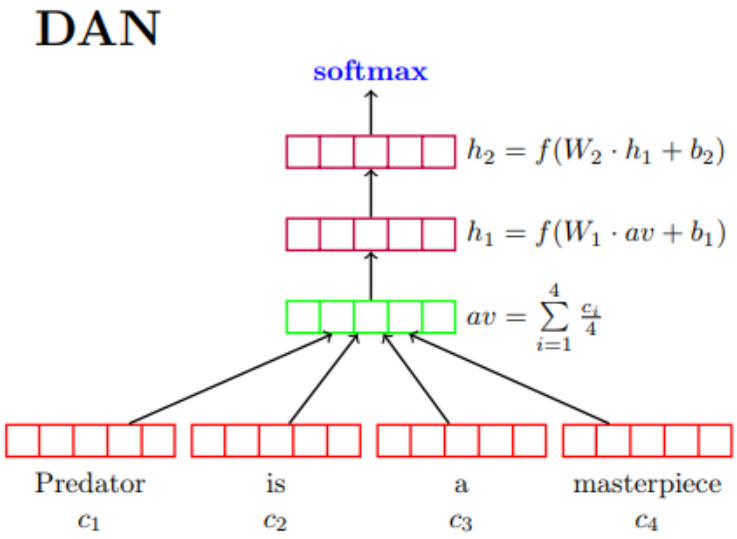
最直观的想法就是忽略句子中词语的顺序，把句子看作是单词的集合。最简单的对所有词嵌入取平均。

好处：模型简单、计算速度快，在处理长文本时有较好的效果。

缺点：忽略了词序，在一些对词序比较敏感的任务（如情感分析）中效果不佳。

DAN

DAN(Deep Averaging Networks)与 Bag of Words 相似，每个 sentence 以词为单位输入，然后直接取平均，然后将网络变 deep，在后面加上几层神经网络。



好处：训练速度和 BOW 几乎相同，神经网络提取出词向量的抽象特征，在情感分析任务中有一定提升。

基于 CNN 的模型

1.《Convolutional neural networks for sentence classification》

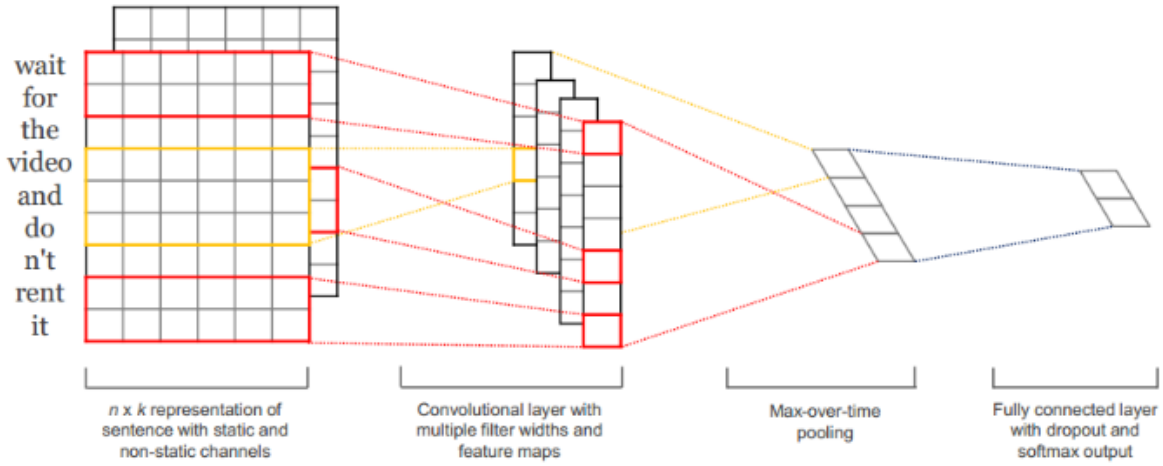


Figure 1: Model architecture with two channels for an example sentence.

输入层是句子中词语对应的词向量依次排列得到的矩阵，假设句子有 n 个词，词向量维度为 k，矩阵的维度是 n×k 的。矩阵可以是静态，就是保持词向量固定不变；也可以是动态的，就是在模型训练过程中将词向量当做可以优化的参数，通常将这种通过 BP 调整词向量值的过程成为 Fine tune。对于未登录词，用 0 或随机小的正数来填充。

输入层通过卷积得到若干 Feature Map，卷积核大小为 h×k，h 表示考虑纵向词语的个数，k 表示词向量的维数，卷积后得到若干列数为 1 的 Feature Map。

池化层采用 Max-over-time pooling 的方法，从之前的一维 Feature Map 中提取最大值，提取最重要的信号，用这种方式解决输入句子长度可变的问题。最终池化层的输出是各个 Feature Map 的最大值组成的一维向量。

池化层得到的一维向量通过全连接的方式连接 Softmax 层，可以在倒数第二层的全连接部分加入 Dropout，防止隐层过拟合。

实验结果如下：

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–	–	–
RNTN (Socher et al., 2013)	–	45.7	85.4	–	–	–	–
DCNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0	–	–
Paragraph-Vec (Le and Mikolov, 2014)	–	**48.7**	87.8	–	–	–	–
CCAE (Hermann and Blunsom, 2013)	77.8	–	–	–	–	–	87.2
Sent-Parser (Dong et al., 2014)	79.5	–	–	–	–	–	86.3
NBSVM (Wang and Manning, 2012)	79.4	–	–	93.2	–	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	–	–	**93.6**	–	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	–	–	93.4	–	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	–	–	**93.6**	–	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	–	–	–	–	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	–	–	–	–	–	82.7	–
SVM$_S$ (Silva et al., 2011)	–	–	–	–	**95.0**	–	–

Table 2: Results of our CNN models against other methods. **RAE**: Recursive Autoencoders with pre-trained word vectors from Wikipedia (Socher et al., 2011). **MV-RNN**: Matrix-Vector Recursive Neural Network with parse trees (Socher et al., 2012). **RNTN**: Recursive Neural Tensor Network with tensor-based feature function and parse trees (Socher et al., 2013). **DCNN**: Dynamic Convolutional Neural Network with k-max pooling (Kalchbrenner et al., 2014). **Paragraph-Vec**: Logistic regression on top of paragraph vectors (Le and Mikolov, 2014). **CCAE**: Combinatorial Category Autoencoders with combinatorial category grammar operators (Hermann and Blunsom, 2013). **Sent-Parser**: Sentiment analysis-specific parser (Dong et al., 2014). **NBSVM, MNB**: Naive Bayes SVM and Multinomial Naive Bayes with uni-bigrams from Wang and Manning (2012). **G-Dropout, F-Dropout**: Gaussian Dropout and Fast Dropout from Wang and Manning (2013). **Tree-CRF**: Dependency tree with Conditional Random Fields (Nakagawa et al., 2010). **CRF-PR**: Conditional Random Fields with Posterior Regularization (Yang and Cardie, 2014). **SVM$_S$**: SVM with uni-bi-trigrams, wh word, head word, POS, parser, hypernyms, and 60 hand-coded rules as features from Silva et al. (2011).

CNN-rand：所有的词向量是随机初始化的，在模型训练过程中同步优化。

CNN-static：所有词向量使用 word2vec 得到的结果，在训练过程中保持不变。

CNN-non-static：所有词向量使用 word2vec 得到的结果且在训练过程中会进行优化。

CNN-multichannel：CNN-static 和 CNN-non-static 混合输入。

结论：

1） CNN-static 较与 CNN-rand 好，说明 pre-training 的 word vector 确实有较大的提升作用（因为 pre-training 的 word vector 显然利用了更大规模的文本数据信息）；

2） CNN-non-static 较于 CNN-static 大部分要好，说明适当的 Fine tune 也是有利的，是因为使得 vectors 更加贴近于具体的任务；

3） CNN-multichannel 较于 CNN-single 在小规模的数据集上有更好的表现，实际上 CNN-multichannel 体现了一种折中思想，即既不希望 Fine tuned 的 vector 距离原始值太远，但同时保留其一定的变化空间。

后续 Ye Zhang 团队基于此模型进行了大量的调参实验的，给出以下结论：

1）由于模型训练过程中的随机性因素，如随机初始化的权重参数，mini-batch，随机梯度下降优化算法等，造成模型在数据集上的结果有一定的浮动，如准确率(accuracy)能达到 1.5%的浮动，而 AUC 则有 3.4%的浮动；

2）词向量是使用 word2vec 还是 GloVe，对实验结果有一定的影响，具体哪个更好依赖于任务本身；

3）Filter 的大小对模型性能有较大的影响，并且 Filter 的参数应该是可以更新的；

4）Feature Map 的数量也有一定影响，但是需要兼顾模型的训练效率；

5）1-max pooling 的方式已经足够好了，相比于其他的 pooling 方式而言；

6） 正则化的作用微乎其微

2.《A convolutional neural network for modelling sentences》

提出一种 DCNN(Dynamic)的网络模型，模型的亮点在于提出了一种全新的动态 pooling 方式。

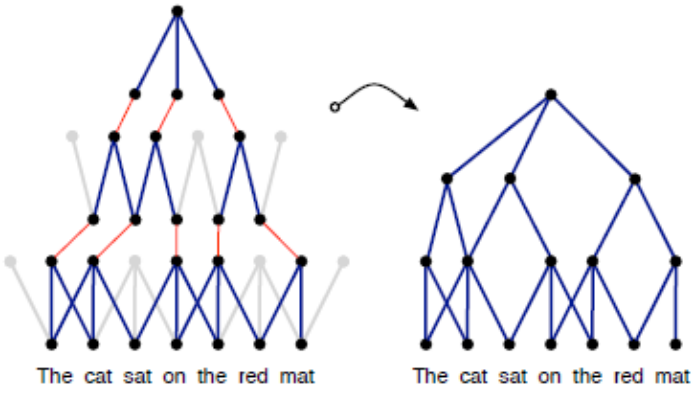


Figure 1: Subgraph of a feature graph induced over an input sentence in a Dynamic Convolutional Neural Network. The full induced graph has multiple subgraphs of this kind with a distinct set of edges; subgraphs may merge at different layers. The left diagram emphasises the pooled nodes. The width of the convolutional filters is 3 and 2 respectively. With dynamic pooling, a filter with small width at the higher layers can relate phrases far apart in the input sentence.

上图是句子语义建模的过程，底层通过组合近邻的词语信息，逐步向上传递，上层由组合新的 Phrase 信息，使得句子中即使距离较远的词也有某种语义联系。总的来说，该模型通过词语间的组合提取出句子中最重要的语义信息。

DCNN 可以处理可变长度的输入，通过网络中的一维卷积层和动态 k-max 池化层。k-max 池化是上一篇文章的最大池化的一般形式。具体来说，k-max pooling 返回的结果不是一个最大值，而是 k 组最大值，这些最大值是原输入的一个子序列；参数 k 可以是一个动态参数。

模型结构如下：

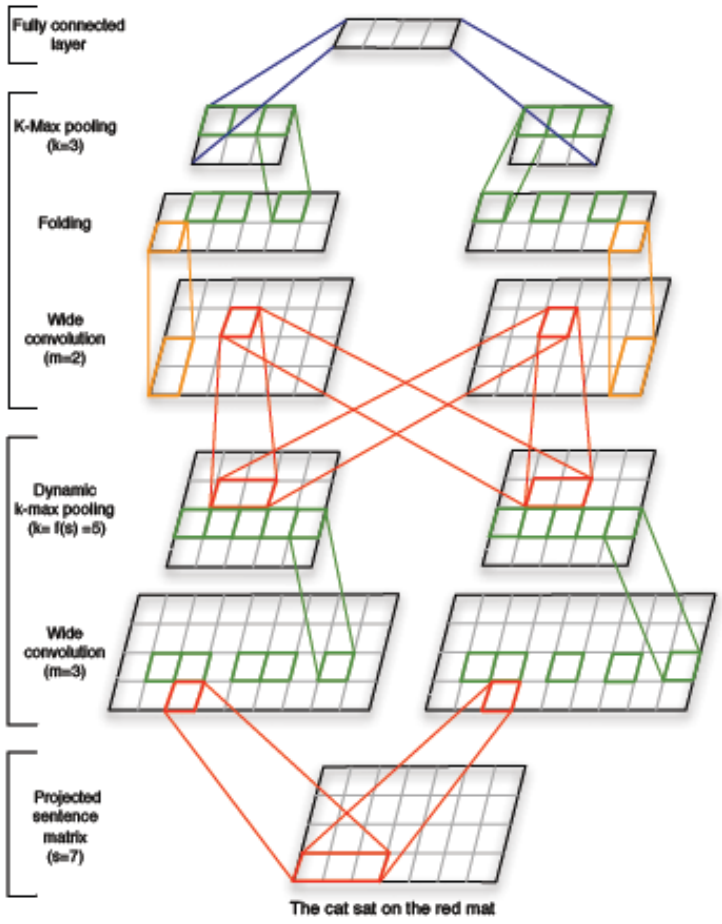


Figure 3: A DCNN for the seven word input sentence. Word embeddings have size $d = 4$. The network has two convolutional layers with two feature maps each. The widths of the filters at the two layers are respectively 3 and 2. The (dynamic) k-max pooling layers have values k of 5 and 3.

网络中的卷积层采用了宽卷积的方式，随后接动态 k-max 池化层，中间的 Feature Map 大小会根据输入句子的长度而变化。具体来说：

1）宽卷积：

与传统卷积相比，宽卷积输出的 Feature Map 的宽度为更宽，因为卷积窗口不需要覆盖所有输入值，可理解为 padding 补零。

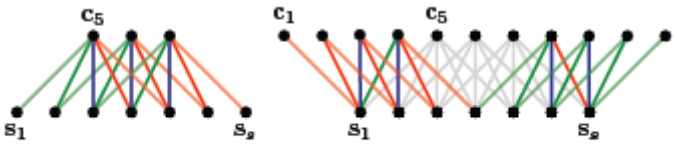


Figure 2: Narrow and wide types of convolution. The filter m has size $m = 5$.

2）k-max 池化

给定一个 k 值和一个序列 p(p≥k)，k-max 池化返回序列 p 中的前 k 个最大值，这些最大值保留原来的次序，是原序列的一个子序列。

通过 k-max 池化既提取了句子中的多个重要信息，同时保留了它们的次序信息。在最后的卷积层上可固定提取 k 个值，通过这种方法允许了不同长度的输入。但对于中间的卷积层，参数 k 不是固定的 。

3）动态 k-max 池化

动态是由可变的 k 值来体现的，k 是句子输入长度和网络深度的一个函数：

$$k_l = \max(\, k_{top},\, \lceil \frac{L-l}{L} s \rceil \,)$$

L 是网络中卷积层层数的总和，l是当前卷积的层数，k_{top}为最顶层卷积层的 pooling 层对应的 k 值，这个值是固定的。动态 k-max 池化的意义在于能够从不同长度的句子中提取出相应数量的语义特征信息。

4）非线性特征函数、多个 Feature Map

与传统 CNN 模型相同。

5）Folding 操作

宽卷积是在输入矩阵d × s的每一行进行计算，d 是词向量的维数，s 是输入句子中的词语数量。Folding 是考虑两行之间的某种联系，通过将两个的向量相加。没有增加参数量的同时，在全连接层之前考虑了特征矩阵中行与行之间的关系。

实验结果如下：

Classifier	Fine-grained (%)	Binary (%)
NB	41.0	81.8
BINB	41.9	83.1
SVM	40.7	79.4
RECNTN	45.7	85.4
MAX-TDNN	37.4	77.1
NBOW	42.4	80.5
DCNN	**48.5**	**86.8**

Classifier	Features	Acc. (%)
HIER	unigram, POS, head chunks NE, semantic relations	91.0
MAXENT	unigram, bigram, trigram POS, chunks, NE, supertags CCG parser, WordNet	92.6
MAXENT	unigram, bigram, trigram POS, wh-word, head word word shape, parser hypernyms, WordNet	93.6
SVM	unigram, POS, wh-word head word, parser hypernyms, WordNet 60 hand-coded rules	95.0
MAX-TDNN	unsupervised vectors	84.4
NBOW	unsupervised vectors	88.2
DCNN	unsupervised vectors	93.0

优势：DCNN 的性能非常好，而且 DCNN 的好处在于不需要任何的先验信息输入，也不需要构造非常复杂的人工特征。

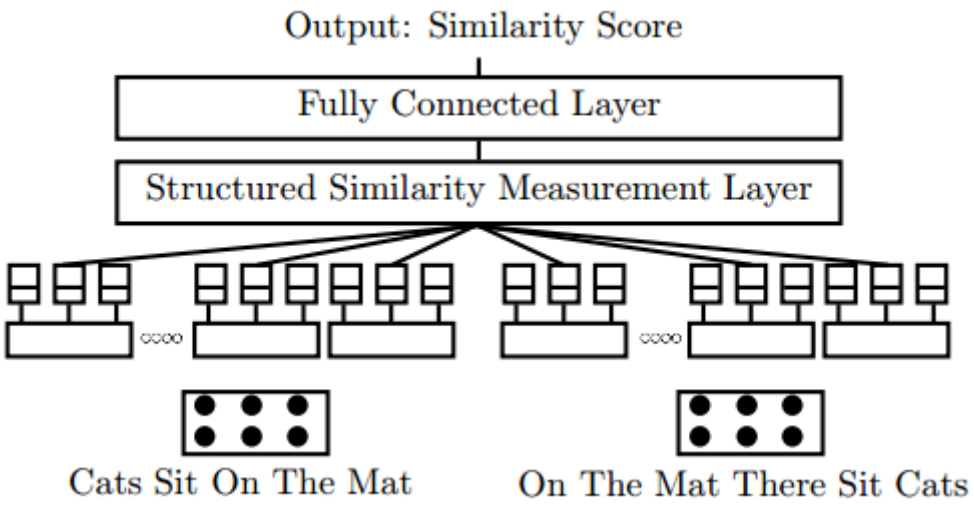
3.《Convolutional neural network architectures for matching natural language sentences》

这篇文章是针对句子匹配问题提出的模型，基础问题还是句子建模，也是基于 CNN 的句子建模网络，后续需要调研句子匹配问题时补充。

4.《Multi-perspective sentence similarity modeling with convolutional neural networks》

这篇论文所提出的模型在所有基于 NN 的模型中在 Paraphrase identification 任务数据集 MSRP 上效果最佳。模型主要分为两个部分，句子的表征模型和相似度计算模型，主要关注一下句子表征模型。

模型的整体框架如下：



句子表征模型：模型是基于 CNN 的，卷积层有两种方式，池化层有三种方式。

1） 卷积层

卷积层有两种方式，分别是粒度为 word 的卷积，和粒度为 embedding 维度上的卷积。

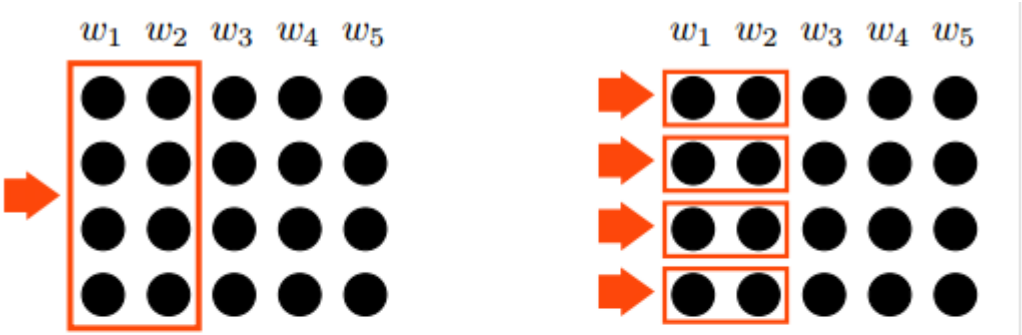


Figure 2: Left: a holistic filter matches entire word vectors (here, $ws = 2$). Right: per-dimension filters match against each dimension of the word embeddings independently.

第一种卷积方式与第一篇 CNN 文章的卷积方式相同，相当于是 n-gram 特征抽取，考虑词与词之间的关系；关于第二种卷积方式，(1)这种方式有助于充分提取输入的特征信息；(2)卷积的粒度更小，在学习过程中的参数调整上每个维度都能得到不同程度的参数调节。PS:第二种卷积方式作者也给不出符合直观想法

的物理意义解释

2） 池化层

模型除了传统的 max-pooling 外还使用了 min-pooling 和 mean-pooling 方式。

作者使用了两种类型 building block，

$$block_A = \{group_A(ws_a, p, sent) : p \in max, min, mean\}$$

$$block_B = \{group_B(ws_b, p, sent) : p \in max, min\}$$

blockA 有三组卷积层，卷积窗口的宽度一致，每一组对应一种池化操作。这里池化操作和卷积层是一一对应的，也就是说并不是一个卷积层上实施三种池化操作。同样的，blockB 有两组卷积层，卷积窗口的宽度一致，两组分别对应 max-pooling 和 min-pooling 的操作。groupB(∗) 中的卷积层对应有 Dim 个以 embedding dimension 为粒度的卷积窗口，也就是对 embedding 的每一维度做卷积运算。

通过组合多样的卷积和池化操作，希望能从多个方面提取输入中的特征信息。

3） 多种窗口尺寸

与传统的 n-gram 类似，在 building block 中使用多种尺寸的卷积窗口，如下图：

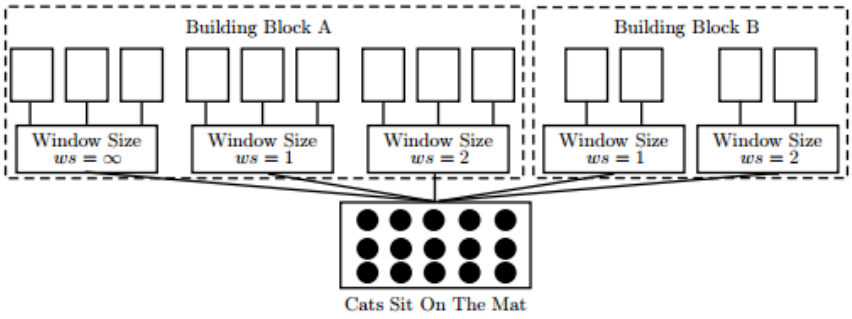


Figure 4: Example neural network architecture for a single sentence, containing 3 instances of $block_A$ (with 3 types of pooling) and 2 instances of $block_B$ (with 2 types) on varying window sizes $ws = 1, 2$ and $ws = \infty$; $block_A$ operates on entire word vectors while $block_B$ contains filters that operate on individual dimensions independently.

ws 表示卷积时卷积的 n-gram 长度，ws 为无穷时表示卷积窗口为整个 word embedding 矩阵。ws 的值和 Feature Map 的数量是需要调参的。

相似度计算模型：

直接在两个句子向量上应用距离度量方法不是最优的，因为在最后生成句子向量中各个部分的意义各不相同，需要对句子向量中的各个部分进行相应的比较和计算。

1）定义了两种相似度计算单元：

$$comU_1(\vec{x}, \vec{y}) = \{\cos(\vec{x}, \vec{y}), L_2Euclid(\vec{x}, \vec{y}), |\vec{x} - \vec{y}|\} \quad (3)$$

$$comU_2(\vec{x}, \vec{y}) = \{\cos(\vec{x}, \vec{y}), L_2Euclid(\vec{x}, \vec{y})\} \quad (4)$$

Cosine distance (cos) measures the distance of two vectors according to the angle between them, while L_2 Euclidean distance ($L_2Euclid$) and element-wise absolute difference measure magnitude differences.

2）基于句子局部的相似度计算

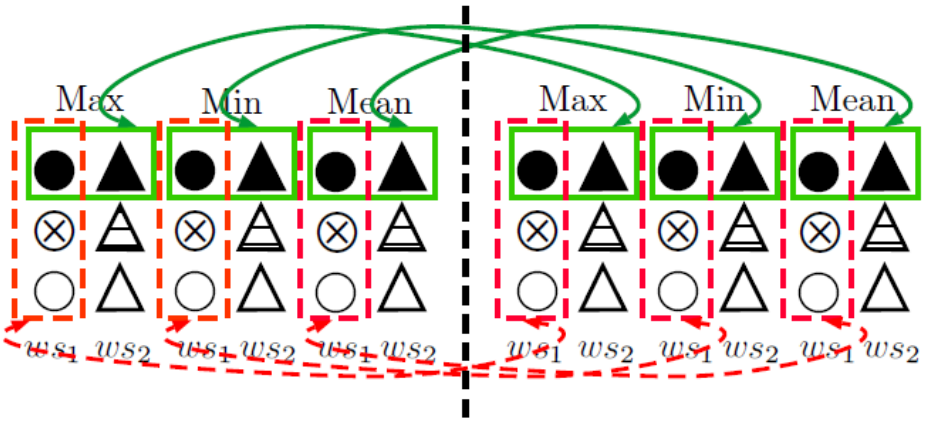


Figure 5: Simplified example of local region comparisons over two sentence representations that use $block_A$ only. The "horizontal comparison" (Algorithm 1) is shown with green solid lines and "vertical comparison" (Algorithm 2) with red dotted lines. Each sentence representation uses window sizes ws_1 and ws_2 with max/min/mean pooling and $numFilter_A = 3$ filters.

基于句子局部的相似度计算之后，得到相应的相似度度量，然后这组向量之后连接一个全连接层，最后 softmax 对应输出，如果是计算相似度度量值，可以用 softmax 输出类别概率值。

MSRP 数据集，用于评测同义句检测 (Paraphrase Identification) 任务的经典数据集，数据集来源于新闻；包含 5801 对句子对，其中 4076 对用于模型训练，而 1725 对用于测试；每一对句子拥有一个标签，0 或者 1,0 表示两个句子不是互为同义句，而 1 则表示两个句子互为同义句。因此这是一个二分类的任务。

$$loss(\theta, x, y_{gold}) = \sum_{y' \neq y_{gold}} \max(0, 1 + f_\theta(x, y') - f_\theta(x, y_{gold}))$$

总结：

文中的模型包含两个部分：卷积-池化模型和相似度计算模型。实验部分验证了模型的有效性，在 MSRP 数据集上模型取得了仅次于 state-of-art 的结果，并且在基于 NN 的方法中是最好的。

模型中的相似度计算层是有必要的，因为对卷积池化处理后的句子成分进行了针对性的比较，从直观上要比直接扔进全连接层更合理。卷积-池化模型单独使用的效果不佳。

二、 Distributed

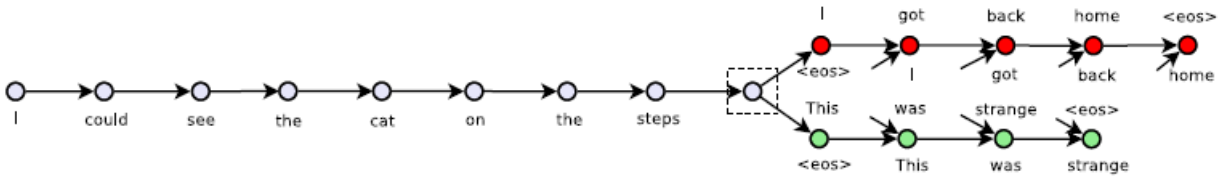
除了以上这种将词向量抽取组合成句向量的方法外，还有一种基于 distributed 的思路，一个 sentence 的表达与上下文的前后 sentence 相关，基于这种思想的工作如下：

1.《Skip-Thought Vectors》

NIPS2015 论文：https://arxiv.org/abs/1506.06726

Github：https://github.com/ryankiros/skip-thoughts

本文旨在提出一个通用的无监督句子表示模型，借鉴了 Word2Vec 中 skip-gram 模型，通过一句话来预测这句话的上一句和下一句，将这个模型称之为 skip-thoughts，模型中抽象出来的过渡向量成为 skip-thought vectors。



$$\sum_t \log P(w_{i+1}^t | w_{i+1}^{<t}, \mathbf{h}_i) + \sum_t \log P(w_{i-1}^t | w_{i-1}^{<t}, \mathbf{h}_i)$$

模型分为两个部分，一个 encoder，两个 decoder，分别 decode 当前句子的上一句和下一句，也就是说，不能预测多个前面和后面的句子。本文采用 GRU-RNN 作为 encoder 和 decoder，将 encoder 部分最后一个词的 hidden state 作为 decoder 的输入。这里用的是最简单的网络结构，并没有考虑复杂的多层网络、双向网络等提升效果。模型中的目标函数也是两个部分，一个来自于预测下一句，一个来自于预测上一句。

这里的 h_i 代表第 i 个句子通过 encoder 的输出，$w_{i+1}^{<t}$代表第 i+1 个句子的前 t-1 个词，基于它们，得到 w_{i+1}^t即第 i+1 个句子的第 t 个词的概率为$\sum_t \log P(w_{i+1}^t | w_{i+1}^{<t}, h_i)$。这个公式的两项分别对应了下一句和上一句出现的概率。本质上，Decoder 部分是作为 Language Modeling 来处理的。

评价：

是一种通用的无监督学习模型，监督学习模型主要是针对具体的下游任务来构造句子向量，仅适用于当前任务并不具有一般性。本文提出的模型采用了 seq2seq 框架，通过搜集大量小说作为训练数据集，模型中得到的 encoder 部分可以作为通用的 feature extractor 给任意句子生成句子向量。对于未登录词的处理，文章

提出了词汇表扩展的方法。

对性能的进一步提升后续可以考虑：

1） 用更深的 encoder 和 decoder 网络

2） 用更大的窗口，不仅仅局限于预测上一句和下一句

3） 尝试将 sentence 扩充到 paragraph

4） 尝试使用不同的 encoder

2.《An efficient framework for learning sentence representations》

Skip-thoughts vectors 的改进版：Quick-thoughts vectors

ICLR 2018 论文：https://openreview.net/pdf?id=rJvJXZb0W

在这项工作中，给定前面一句话来预测下一句话的任务被定义为一个分类任务：解码器被一个分类器所取代，该分类器在一组候选句中选择下一句。在编码之后直接输出给一个分类器，判断哪些是句子的近邻句，哪些不是，算法效率上会高很多。通过 Encoder 对目标句子进行编码，分析目标句子的内在含义并预测相邻句子的含义，模型并不会直接生成近似句，而是在候选句子中选择一个最相近的句子，完成这样一个分类任务。模型倾向于忽略与构建语义特征无关的各种因素，将损失函数侧重于特征空间而不是原始的数据空间。

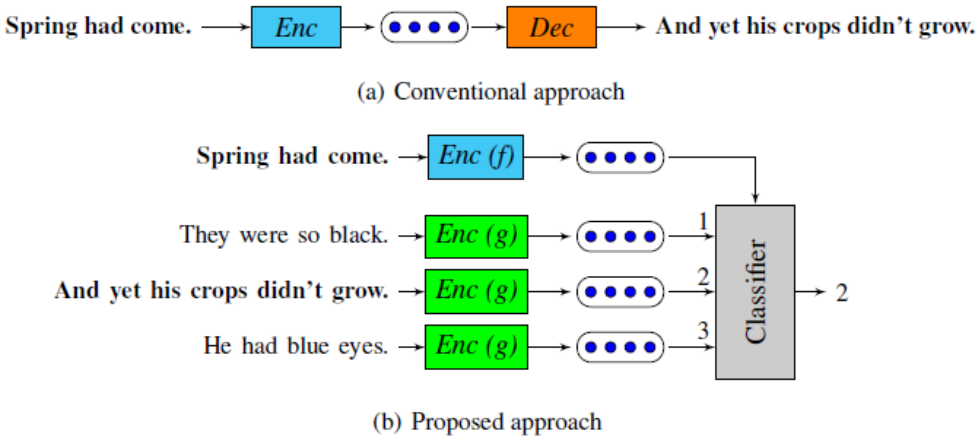


Figure 1: Overview. (a) The approach adopted by most prior work where given an input sentence the model attempts to generate a context sentence. (b) Our approach replaces the decoder with a classifier which chooses the target sentence from a set of candidate sentences.

$$p(s_{\text{cand}}|s, S_{\text{cand}}) = \frac{\exp[c(f(s), g(s_{\text{cand}}))]}{\sum_{s' \in S_{\text{cand}}} \exp[c(f(s), g(s'))]}$$

$$\sum_{s \in D} \sum_{s_{\text{ctxt}} \in S_{\text{ctxt}}} \log p(s_{\text{ctxt}}|s, S_{\text{cand}})$$

f 和 g 是 Encoder，s 为中心句，S_{ctxt} 是 context sentences，S_{cand} 是候选句，它包括 S_{ctxt} 和不相关的句子；c 是 classifier，函数形式与 word2vec 一致，$C(u,v) = u^T v$，与 Skip-Thought 相比，QT 用判别式问题取代了生成式问题，方法更简单训练时间大大缩短。f 和 g 参数独立，在测试阶段，给定句子 s，它的句向量表示为 f(s)和 g(s)的直连。

Model	Dim	Training time (h)	MR	CR	SUBJ	MPQA	TREC	MSRP (Acc)	MSRP (F1)	SICK r	SICK ρ	SICK MSE
GloVe BoW	300	-	78.1	80.4	91.9	87.8	85.2	72.5	81.1	0.764	0.687	0.425
Trained from scratch on BookCorpus data												
SDAE	2400	192	67.6	74.0	89.3	81.3	77.6	**76.4**	83.4	N/A	N/A	N/A
FastSent	<500	2*	71.8	78.4	88.7	81.5	76.8	72.2	80.3	N/A	N/A	N/A
ParagraphVec	<500	4*	61.5	68.6	76.4	78.1	55.8	73.6	81.9	N/A	N/A	N/A
uni-skip	2400	336	75.5	79.3	92.1	86.9	91.4	73.0	81.9	0.848	0.778	0.287
bi-skip	2400	336	73.9	77.9	92.5	83.3	89.4	71.2	81.2	0.841	0.770	0.300
combine-skip	4800	336†	76.5	80.1	**93.6**	87.1	**92.2**	73.0	82.0	0.858	0.792	0.269
combine-cnn	4800	-	77.2	80.9	93.1	**89.1**	91.8	75.5	82.6	0.853	0.789	0.279
uni-QT	2400	11	77.2	82.8	92.4	87.2	90.6	74.7	82.7	0.844	0.778	0.293
bi-QT	2400	9	77.0	83.5	92.3	87.5	89.4	74.8	82.9	0.855	0.787	0.274
combine-QT	4800	11†	**78.2**	**84.4**	93.3	88.0	90.8	76.2	**83.5**	**0.860**	**0.796**	**0.267**
Trained on BookCorpus, pre-trained word vectors are used												
combine-cnn	4800	-	77.8	82.1	93.6	**89.4**	92.6	76.5	83.8	0.862	0.798	0.267
MC-QT	4800	11	**80.4**	**85.2**	**93.9**	**89.4**	**92.8**	**76.9**	**84.0**	**0.868**	**0.801**	**0.256**
Trained on (BookCorpus + UMBC) data, from scratch and using pre-trained word vectors												
combine-QT	4800	28	81.3	84.5	94.6	89.5	92.4	75.9	83.3	0.871	0.807	0.247
MC-QT	4800	28	**82.4**	**86.0**	**94.8**	**90.2**	92.4	**76.9**	**84.0**	**0.874**	**0.811**	**0.243**

在大多数情况下，uni/bi/combine-QT 的表现好于其他模型，并且训练时间大大缩短。uni-QT 和 bi-QT 分别指使用了单向和双向 RNN 作为 encoder 的模型。combine-QT 在 test time 使用了 uni-QT 和 bi-QT 学习得到的句子表征(由于它同时使用了两种表征，我们可以看到在 dim 一栏值是其它值的两倍)。在这里 uni/bi/combine-QT 都没有使用 pre-trained word vectors。

注意 MC-QT 模型在各个任务上最优。MC-QT 就是 MultiChannel-QT，作者模仿了 Yoon Kim 在 Convolutional neural networks for sentence classification(2014) 中的做法，分别用一个 pretrained word embedding(Glove)和一个 tunable word embedding 来作为输入。

评价：

该模型的优势在于训练速度，分类任务比生成相似句子要快一个数量级，对于处理大规模数据集有实际意义，采用简单的分类器迫使 encoder 学习更佳的句向量表示。

三、简单的改进

Bag-of-Words 简单加强版：

1.《A Simple but Tough-to-beat Baseline for Sentence Embeddings》

ICLR 2017 conference submission

论文：https://openreview.net/forum?id=SyK00v5xx

github：https://github.com/PrincetonML/SIF

本文来自普林斯顿大学， 随着词向量在 NLP 任务中的成功， 更多的人开始探索较长文本如短语， 句子，段落的向量表示，本文提出了一种简单但是有效的句向量算法，只需要将句子中的向量进行加权平均，再减去句子的矩阵的主成分即可。但是本文算法在句子相似度，文本蕴含，文本分类问题都取得了更好的效果，甚至在部分任务上，超过了监督方法训练的句向量。

作者将该算法称之为 WR。 选择热门词嵌入技术，通过线性加权组合对一个句子进行编码，在移除它们的第一主成分上的矢量。

W 表示 Weighted，意为使用预估计的参数给句中的每个词向量赋予权重。

R 表示 Removal，意为使用 PCA 或者 SVD 方法移除句向量中的无关部分。

Algorithm 1 Sentence Embedding

Input: Word embeddings $\{v_w : w \in \mathcal{V}\}$, a set of sentences $\mathcal{S}$, parameter a and estimated probabilities $\{p(w) : w \in \mathcal{V}\}$ of the words.
Output: Sentence embeddings $\{v_s : s \in \mathcal{S}\}$

1: **for all** sentence s in $\mathcal{S}$ **do**
2: $\quad v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$
3: **end for**
4: Form a matrix X whose columns are $\{v_s : s \in \mathcal{S}\}$, and let u be its first singular vector
5: **for all** sentence s in $\mathcal{S}$ **do**
6: $\quad v_s \leftarrow v_s - uu^\top v_s$
7: **end for**

输入：

预训练的词向量；待处理的句子集合 S；参数 a；词频估计

输出：

句子向量

第一步：对句子的每个词向量乘以一个权值，这个权值是一个常数 α 除以 α 与该词语频率的和，使得高频词的权值会相对下降，求和后得到暂时的句子向量。

第二步：计算语料库所有句子向量构成的矩阵的第一个主成分 u，让每个句子向量减去它在 u 上的投影，类似 PCA。

结果分析：

	Results collected from (Wieting et al. 2016) except tfidf-GloVe											Our approach	
Supervised or not	Su.							Un.			Se.	Un.	Se.
Tasks	PP	PP -proj.	DAN	RNN	iRNN	LSTM (no)	LSTM (o.g.)	ST	avg-GloVe	tfidf-GloVe	avg-PSL	GloVe +WR	PSL +WR
STS'12	58.7	**60.0**	56.0	48.1	58.4	51.0	46.4	30.8	52.5	58.7	52.8	56.2	59.5
STS'13	55.8	56.8	54.2	44.7	56.7	45.2	41.5	24.8	42.3	52.1	46.4	56.6	**61.8**
STS'14	70.9	71.3	69.5	57.7	70.9	59.8	51.5	31.4	54.2	63.8	59.5	68.5	**73.5**
STS'15	75.8	74.8	72.7	57.2	75.6	63.9	56.0	31.0	52.7	60.6	60.0	71.7	**76.3**
SICK'14	71.6	71.6	70.7	61.2	71.2	63.9	59.0	49.8	65.9	69.4	66.4	72.2	**72.9**
Twitter'15	52.9	52.8	**53.7**	45.1	52.9	47.6	36.1	24.7	30.3	33.8	36.3	48.0	49.0

	PP	DAN	RNN	LSTM (no)	LSTM (o.g.)	skip-thought	Ours
similarity (SICK)	84.9	85.96	73.13	85.45	83.41	85.8	**86.03**
entailment (SICK)	83.1	84.5	76.4	83.2	82.0	-	**84.6**
sentiment (SST)	79.4	83.4	86.5	86.6	**89.2**	-	82.2

在句子相似度任务上超过平均水平，还超过部分复杂的模型，在句子分类上效果明显。

2.《Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms》

ACL 2018 paper

论文：https://arxiv.org/abs/1805.09843

Github：https://github.com/dinghanshen/SWEM

文章重新审视了 deep learning models 在各类 NLP 任务中的必要性，通过 17 个数据集的实验探究，论文详细比较了直接在词向量上进行池化的简单模型和主流神经网络模型在 NLP 多个任务上的效果，实验结

果表明，在很多任务上简单的词向量模型和神经网络模型效果相当，有些任务甚至简单模型更好。

在 NLP 领域，目前代表性的词向量工作如 word2vec 和 GloVe，使用词向量将一个变长文本表示为一个固定向量的常用方法有 a) 以词向量为输入，使用一个复杂的神经网络来进行文本表示学习；b) 在词向量的基础上，直接简单的使用按元素求均值或相加的简单方法来表示。

该文对比的主流神经网络模型为 LSTM 和 CNN。LSTM 的特征在于使用门机制来学习长距离依赖信息，可以认为考虑了词序信息。CNN 利用滑动窗口卷积连续词的特征，然后通过池化学习到最显著的语义特征，对于简单的词向量模型(simple word-embedding model, SWEM)作者提出了下面几种方法：

1) Average Pooling

$Z = \frac{1}{L}\sum_{i=1}^{L} v_i$ 对词向量安元素求均值，相当于考虑每个词的信息。

2) Max Pooling

$Z = \mathrm{Max} - \mathrm{pooling}(v_1, v_2, \ldots, v_L)$ 对词向量每一维取最大值，相当于考虑最显著的特征信息。

3) Concat

对上面两种池化方法得到的结果拼接。

4) Hierarchical Pooling

上面的方法没有考虑到词序和空间信息，提出层次池化。先使用大小为 n 的局部窗口进行平均池化，再使用全局最大池化，该方法类似 n-grams 特征。

对比 SWEM 和神经网络模型，SWEM 仅对词向量使用了池化操作，并没有额外的参数，且可以高度并行化。

Model	Parameters	Complexity	Sequential Ops
CNN	$n \cdot K \cdot d$	$\mathcal{O}(n \cdot L \cdot K \cdot d)$	$\mathcal{O}(1)$
LSTM	$4 \cdot d \cdot (K + d)$	$\mathcal{O}(L \cdot d^2 + L \cdot K \cdot d)$	$\mathcal{O}(L)$
SWEM	0	$\mathcal{O}(L \cdot K)$	$\mathcal{O}(1)$

Table1：Comparison of CNN, LSTM and SWEM architectures

实验结果分析：

实验中使用了 300 维的 GloVe 词向量，对未登录词按均匀分布进行初始化，最终的词嵌入通过训练一个多层感知机 MLP 来得到，用 Adam 算法来优化模型。在文本分类，文本序列匹配和句子分类三大任务共 17 个数据集上进行实验分析。

文档分类:

实验中的文档分类是任务能被分为三种类型：主题分类，情感分类和本体分类，实验结果如 Table2:

Model	Yahoo! Ans.	AG News	Yelp P.	Yelp F.	DBpedia
Bag-of-means*	60.55	83.09	87.33	53.54	90.45
Small word CNN*	69.98	89.13	94.46	58.59	98.15
Large word CNN*	70.94	91.45	95.11	59.48	98.28
LSTM*	70.84	86.06	94.74	58.17	98.55
Deep CNN (29 layer)†	73.43	91.27	**95.72**	**64.26**	**98.71**
fastText ‡	72.0	91.5	93.8	60.4	98.1
fastText (bigram)‡	72.3	92.5	95.7	63.9	98.6
SWEM-*aver*	73.14	91.71	93.59	60.66	98.42
SWEM-*max*	72.66	91.79	93.25	59.63	98.24
SWEM-*concat*	**73.53**	**92.66**	93.76	61.11	**98.57**
SWEM-*hier*	73.48	92.48	**95.81**	**63.79**	98.54

Table2: Test accuracy on (long) document classification tasks

在主题分类任务中（第一第二个任务），SWEM 模型取得了比深度学习模型更好的结果，尤其是 SWEM-concat 模型效果优于 29 层的 Deep CNN。在本体分类(DBpedia)上也有类似的趋势。在情感分析任务中，深度学习模型要优于不考虑词序信息的 SWEM 模型。考虑了词序信息和空间信息的 SWEM-hier 取得了和 CNN/LSTM 相当的结果。这主要是优于情感分析任务需要把握词序信息。

在大多数任务上 SWEM-max 的方法略差于 SWEM-aver，但是它提供了互补的信息，所以 SWEM-concat 获得了更好的结果。值得一提的是，SWEM-max 学得的词嵌入比 GloVe 词向量更加稀疏。这表明模型可能仅依赖于整个词汇中的几个关键词进行预测，通过嵌入，模型可以学得特定任务的重要单词。

Politics	Science	Computer	Sports	Chemistry	Finance	Geoscience
philipdru	coulomb	system32	billups	sio2 (SiO_2)	proprietorship	fossil
justices	differentiable	cobol	midfield	nonmetal	ameritrade	zoos
impeached	paranormal	agp	sportblogs	pka	retailing	farming
impeachment	converge	dhcp	mickelson	chemistry	mlm	volcanic
neocons	antimatter	win98	juventus	quarks	budgeting	ecosystem

Table3: Top five words with the largest values in a given word-embedding dimension

论文在整个词汇表中根据词向量维度挑选出了一个维度中值最大的 5 个词展示在 Table3 中。可以看到每个维度选出的词是同一个主题相关的。甚至模型可以学到没有标签信息的结构，例如 Table3 中的“Chemistry”，在数据集中是没有 chemistry 标签的，它属于 science 主题。

在时间复杂度上，SWEM 模型也比 CNN 和 LSTM 模型更加高效。

文本序列匹配：

在句子匹配问题的实验中，主要包括自然语言推理，问答中答案句选择和释义识别任务。实验结果如下：

Model	SNLI	MultiNLI		WikiQA		Quora	MSRP	
		Matched	Mismatched					
	Acc.	*Acc.*	*Acc.*	*MAP*	*MRR*	*Acc.*	*Acc.*	*F1*
CNN	82.1	65.0	65.3	0.6752	0.6890	79.60	69.9	80.9
LSTM	80.6	66.9*	66.9*	**0.6820**	**0.6988**	82.58	70.6	80.5
SWEM-*aver*	82.3	66.5	66.2	**0.6808**	**0.6922**	82.68	71.0	81.1
SWEM-*max*	**83.8**	**68.2**	**67.7**	0.6613	0.6717	82.20	70.6	80.8
SWEM-*concat*	83.3	67.9	67.6	0.6788	0.6908	**83.03**	**71.5**	**81.3**

可以看到除了 WikiQA 数据集，其他数据集上，SWEM 模型获得了比 CNN 和 LSTM 更好的结果。这可能是因为在当匹配自然语言句子时，在大多数情况下，只需要使用简单模型对两个序列之间在单词级别上进行对比就足够了。从这方面也可以看出，词序信息对于句子匹配的作用比较小。此外简单模型比 LSTM 和 CNN 更容易优化。

句子分类：

相比于前面的文档分类，句子分类任务平均只有 20 词的长度，实验结果如下：

Model	MR	SST-1	SST-2	Subj	TREC
RAE (Socher et al., 2011b)	77.7	43.2	82.4	–	–
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–
LSTM (Tai et al., 2015)	–	46.4	84.9	–	–
RNN (Zhao et al., 2015)	77.2	–	–	**93.7**	90.2
Constituency Tree-LSTM (Tai et al., 2015)	-	**51.0**	88.0	-	-
Dynamic CNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0
CNN (Kim, 2014)	**81.5**	48.0	**88.1**	93.4	**93.6**
DAN-ROOT (Iyyer et al., 2015)	-	46.9	85.7	-	-
SWEM-*aver*	77.6	45.2	83.9	92.5	**92.2**
SWEM-*max*	76.9	44.1	83.6	91.2	89.0
SWEM-*concat*	**78.2**	**46.1**	**84.3**	**93.0**	91.8

Table5: Test accuracies with different compositional functions on (short) sentence classifications

在情感分类任务上，和前面文档分类的实验结果一样，SWEM 效果差于 LSTM 和 CNN，在其他两个任务上，效果只是略差于 NN 模型。相比与前面的文档分类，在短句子分类上 SWEM 的效果要比长文档的分类效果要差。这也可能是由于短句中词序信息更重要。

词序信息的重要性:

从上面可以看到，SWEM 模型的一个缺点在于忽略了词序信息，而 CNN 和 LSTM 模型能够一定程度的学习词序信息。那么在上述的这些任务中，词序信息到底有多重要？

为了探索这个问题，该文将训练数据集的词序打乱，并保持测试集的词序不变，就是为了去掉词序信息。然后使用了能够学习词序信息 LSTM 模型进行了实验，实验结果如下：

Datasets	Yahoo	Yelp P.	SNLI
Original	72.78	95.11	78.02
Shuffled	72.89	93.49	77.68

在 Yahoo 和 SNLI 数据集（也就是主题分类和文本蕴涵任务）上，在乱序训练集上训练的 LSTM 取得了和原始词序相当的结果。这说明词序信息对这两个问题并没有明显的帮助。但是在情感分析任务上，乱序的 LSTM 结果还是有所下降，说明词序对于情感分析任务还是比较重要。

再来看看 SWEM-hier 在情感分析上的效果，相比与 SWEM 其他模型，SWEM-hier 相当于学习了 n-gram 特征，保留了一定的局部词序信息。在两个情感任务上效果也是由于其他 SWEM 模型，这也证明了 SWEM-hier 能够学习一定的词序信息。

总结：

该论文展示了在词向量上仅使用池化操作的简单模型 SWEM 的性能，在多个 NLP 任务数据集上进行了实验，比较了 SWEM 和目前主流的 NN 模型（CNN 和 LSTM）性能。

实验发现，SWEM 这样简单的基线系统在很多任务上取得了与 NN 相当的结果：

1. 简单的池化操作对于长文档（上百个词）表示具有不错的表现，而循环和卷积操作对于短文本更有效；

2. 情感分析任务相比主题文本分类任务对词序特征更敏感，但是该文提出的一种简单的层次池化也能够学习一定的词序信息，和 LSTM/CNN 在情感分析任务上取得了相当的结果；

3. 对于句子匹配问题，简单的池化操作已经展现出了与 LSTM/CNN 相当甚至更好的性能；

4. 对于 SWEM-max 模型，可以通过对词向量维度的分析得到较好的模型解释；

5. 在一些任务上，词向量的维度有时在低维已经足够好；

6. 在标注训练集规模小的时候，简单的 SWEM 模型可能更加鲁棒、获得更好的表现。

总的来说，我们在进行研究时，有时为了让模型学习到更为丰富的信息，会把模型设计得十分复杂，但是这篇论文通过实验告诉了我们，简单的基线系统也能够获得很不错的表现。当我们做具体任务时，应该根据具体需求来选择设计模型，简单有效的系统也应该受到关注。

监督学习的句向量表示一直被认为比无监督学习的通用性差，直到 InferSent 的提出

3.《Supervised Learning of Universal Sentence Representations from Natural Language Inference Data》

与前面提到的无监督方法不同，监督学习需要给数据集打标来标注某些任务，如自然语言推理或机器翻译，它们也引出了对于具体任务的倾向问题，以及高质量嵌入表示所需的数据集大小的相关问题。

因其简单的体系结构，使得 InferSent 成为一个非常有趣的方法。 它使用 Sentence Natural Language Inference（NLI）数据集（包含 570k 对标有 3 种类别的句子：中性，矛盾和包含）来在句子编码器之上训练分类器。 两个句子都使用相同的编码器进行编码，而分类器则是根据两个句嵌入构建的句向量对进行训练。

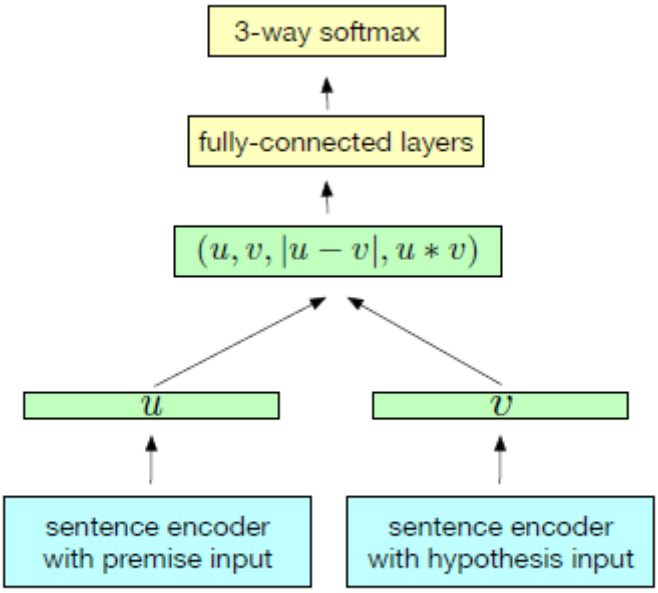


Figure 1: **Generic NLI training scheme.**

训练过程：

句子对中的每一个句子都传给同样一个 sentence encoder, 这个 encoder 产生的 sentence embeddings(图中的 u 和 v)要经过 3 种匹配函数：连接，element-wise 的向量差(取绝对值)，element-wise 的向量积。在这之后经过全连接层传给 3-way softmax 进行句子关系的预测。

作者尝试了 7 种不同的编码器模型：

1. standard recurrent encoders with LSTM
2. standard recurrent encoders with GRU

这两种是基础的 RNN encoder，将网络中最后一个隐藏层状态作为 sentence representation；

3. conncatenation of last hidden states of forward and backward GRU

这种方法是将单向的网络变成了双向的网络，然后用将前向和后向的最后一个状态进行连接，得到句子向量

4. Bi-directional LSTMs with mean pooling
5. Bi-directional LSTMs with max pooling

这两种方法使用了双向 LSTM 结合一个 pooling 层的方法来获取句子表示，具体如下：

$$
\begin{aligned}
\overrightarrow{h_t} &= \overrightarrow{\text{LSTM}_t}(w_1, \ldots, w_T) \\
\overleftarrow{h_t} &= \overleftarrow{\text{LSTM}_t}(w_1, \ldots, w_T) \\
h_t &= [\overrightarrow{h_t}, \overleftarrow{h_t}]
\end{aligned}
$$

按照公式得到每个时刻 t 的隐藏状态 h_t 后，经过一个 max/mean pooling 得到最终的句子表示。网络模型如下：

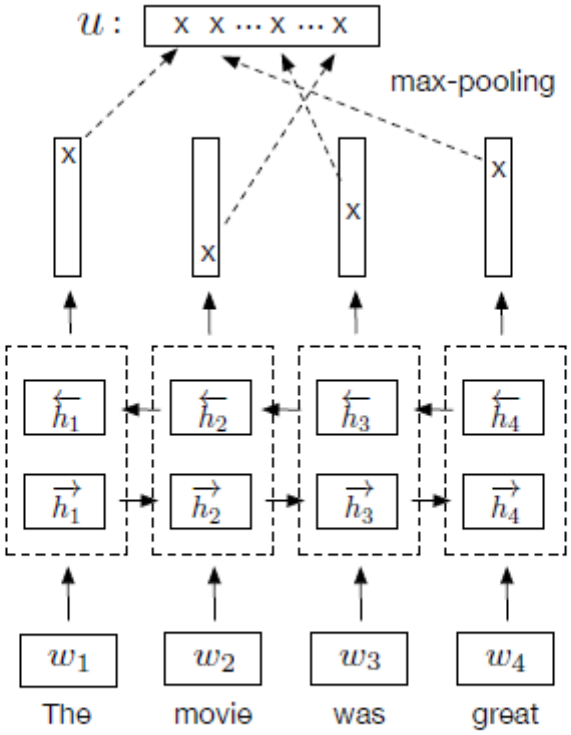


Figure 2: **Bi-LSTM max-pooling network.**

6. self-attentive network

在双向 LSTM 中加入了 attention 机制，网络结构如下：

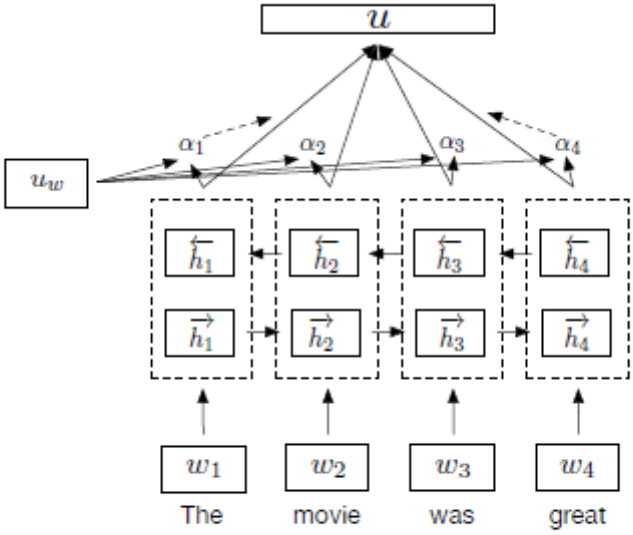


Figure 3: **Inner Attention network architecture.**

BiLSTM 的输出首先经过一次线性变换和一次非线性 tanh 变换，然后计算每个隐藏状态对应的注意力权重，最后加权求和得到最终表示。来找到隐向量的 weight，找到更好的 sentence embedding。

7. hierarchical convolutional networks

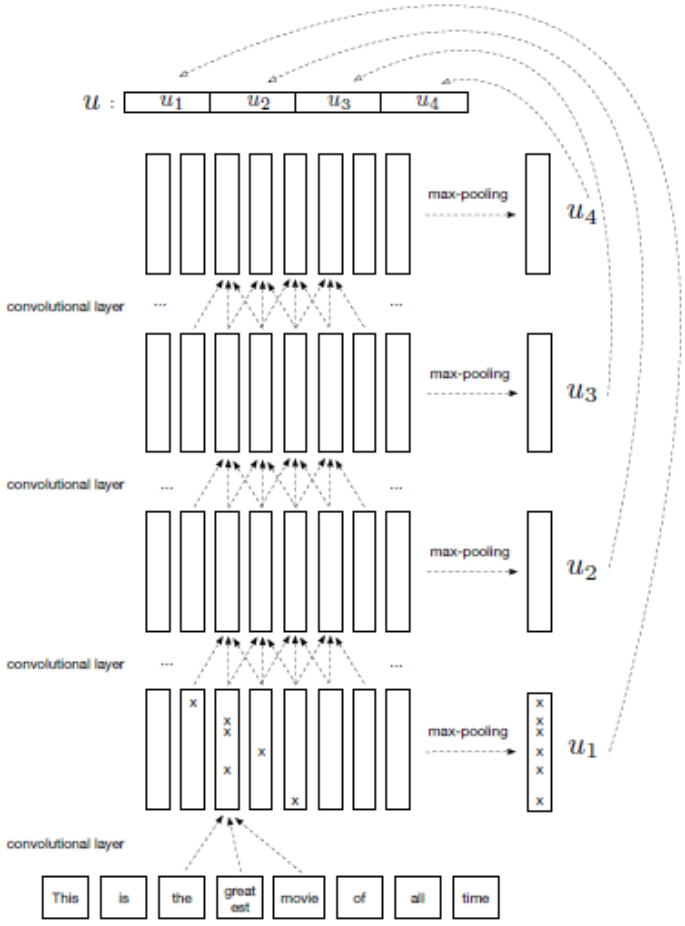


Figure 4: **Hierarchical ConvNet architecture.**

通过多层卷积神经网络可以对输入句子进行不同层级的抽象，在每一层作者通过 max-pooling 得到一个表示 u_i，最终将这些表示连接得到句向量表示。

BiLSTM 由于包含了两个方向的信息，效果好于 LSTM/GRU。然后，ConvNet 强于抓取 local 的信息但缺点是不能很好地概括句子中 global 的信息，在 NLP 任务中往往不如 LSTM 类的模型。attention+(Bi)LSTM 效果要强于(Bi)LSTM：因为有了 attention 机制，我们可以找到每个隐向量在任务中合适的权重，而不是简单的对隐向量做简单的平均。但是，这里找到的对 NLI 任务有用的 weights，并不一定能很好地迁移到其他任务中去。也就是说，这里的 attention 机制对导致其在训练任务中表现的好，但是在 transfer 任务中表现一般。最终，论文采用了一个 max-pooling 的 bi-directional LSTM 作为编码器。

评价：

文章引起了对寻找 NLP 领域的 ImageNet 和 VGG-Net 的讨论。虽然 sentence embedddings 是基于 supervised 任务产生的，可是一样可以很好的 transfer 到其他任务中去。这给很多缺乏训练数据的 NLP 任务带来了方便。为后续的多任务学习和迁移学习提供了启发。

4.《Learning General Purpose Distributed Sentence Representations via Large Scale Multi-task Learning》

什么样的监督训练任务能够学习到提升下游任务效果的句子嵌入？

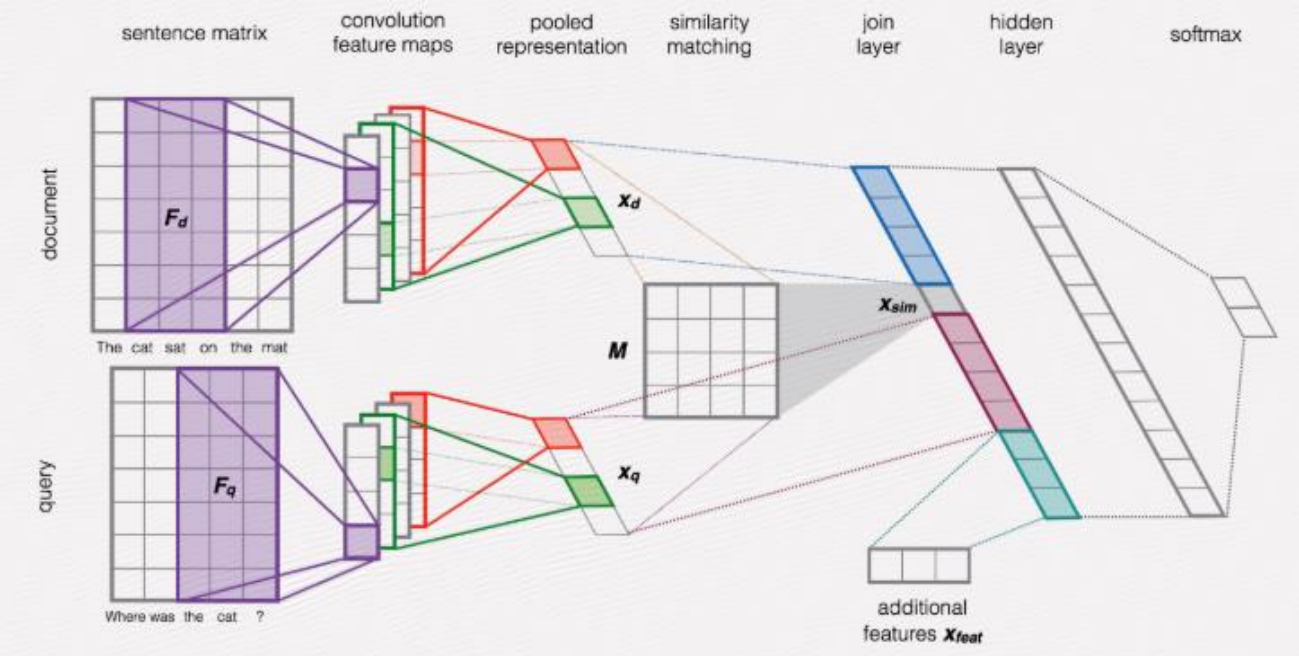
尝试在训练方案中结合多个训练目标。为学得一个较好的通用句子嵌入，编码需考虑到同一个句子的多个方面。作者利用一对多的多任务学习框架，通过在多个任务间切换来学习通用句嵌入。选择 6 个任务，共享优化双向 GRU 获得的相同句嵌入。

同样思路的还有 Google 的通用句子编码器，编码器经过各种数据源和各种任务的训练，以动态地适应各种自然语言处理任务，已开放 TF 版本的预训练编码器。

四、项目相关

1. 基于 Siamese 结构的神经网络模型

先分别学习输入的文本对的句子向量表达，再基于句子向量计算相似度。常见于 QA 系统。



采用上下并行的相同参数的 CNN(简单的 SWEM 或任意 Decoder)学习输入的问题和答案的句子向量表达，然后经过相似度矩阵 M 计算相似度，全连接层和隐藏层进行特征整合和非线性变换，最后 Softmax 层来输出输入候选答案被预测为正确或者错误的概率。

同样的结构结合跳跃卷积、K-max 采样是否应用到我们的相似句匹配？前后句连接可以转化为找前一句的相似句吗？

2. 《Learning Semantic Textual Similarity from Conversations》从对话中学习

论文提出 Input-Response Prediction 的假设，在对话中如果两个问句的回答是相似的，那么这两个问句的相似度就高。算法的任务是从一堆候选句子中找到给定问句的正确答案。

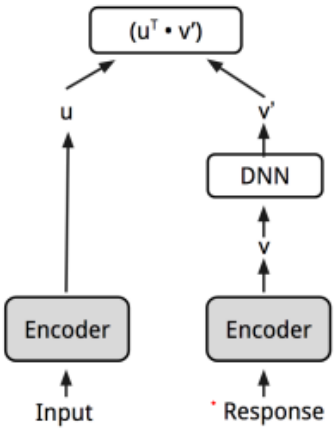
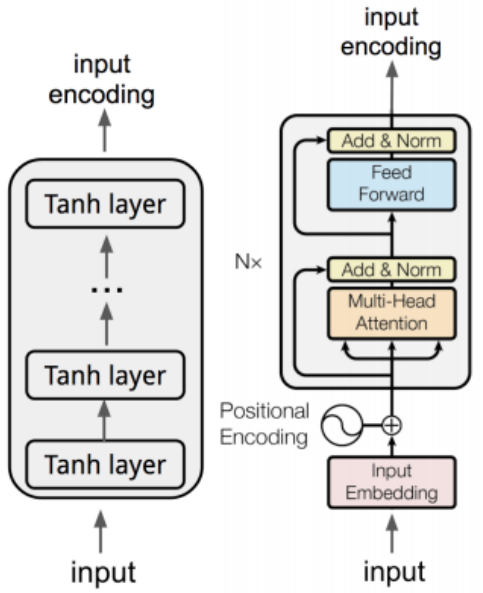


Figure 3: The basic input-response encoder model. The sentence encoders in gray boxes use shared parameters. The DNN performs the mapping between the semantics of the input sentence and the expected response.



(a) DAN encoder (b) Transformer encoder

Figure 4: Model architectures for the DAN and Transformer sentence encoders.

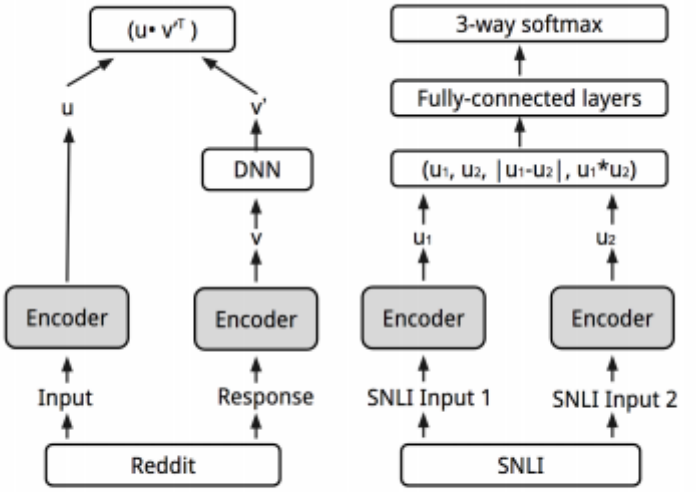


Figure 5: The architecture of the multitask model. The encoders in gray boxes use shared parameters.

注意这里并不是一个对称的图形，为了反映输入(input)和回答(response)的不同，response embedding 传

给了 DNN，经过一系列操作后产生的新的 embedding 和 input 这边的 embedding 做点积。通过 Softmax 将点积的结果转换为概率，训练模型参数以最大化正确 Response 的对数似然。

图中 Input 和 Response 的 Encoder 是共享的。作者选取了 DAN 和 Transformer。二者的差异在于后者精度更高，当然内存和时间方面的损耗也更高。作为输入的 embedding 全部是在训练中学习得来。

作者认为加入多任务学习以后模型应该能学习到更好的语义表征。原因是多任务可以覆盖到更多的语义现象，而这在单任务中是无法学习到的。于是作者在原本的 conversational input-response prediction 的基础上增加了自然语言推断任务(NLI)，对应的数据是 SNLI。第二个任务的模型和前面提到的 InferSent 是一模一样的。两个任务共享 encoder。

3. 《Concatenated p-mean Word Embeddings as Universal Cross-Lingual Sentence Representations》

简单有效的非深度模型，作者把“对 word embedding 求平均”的操作泛化为 p-mean 的一类操作，进而推广到使用不同的 p 值产生不同的特征。

具体的，power-means 定义为：

$$\left(\frac{x_1^p+\cdots+x_n^p}{n}\right)^{1/p} \quad ; \quad p \in \mathbb{R} \cup \{\pm\infty\}$$

当 p=1 时，它就是取平均的操作。另外，当 p=+∞，它是取最大(max)的操作，当 p=−∞时，它是取最小值(min)的操作。作者实验了不同的 p 值，结论是以上三种操作（平均最大和最小值）放在一起使用效果会非常好。

给定一个句子的 word embeddings(假设有 n 个词，每个 embedding 有 d 维)：

$$W = [w_1, \ldots, w_n] \in R^{n \times d}$$

用 $H_p(W)$表示对其进行 p-means 操作的结果。那么不同的 p-means 操作的结果可以连接起来表达为：

$$\mathbf{s}^{(i)} = H_{p_1}(\mathbf{W}^{(i)}) \oplus \cdots \oplus H_{p_K}(\mathbf{W}^{(i)})$$

仅仅对一种 embedding 采取不同的 power-means 还不够。作者使用了多种 embeddings(Word2Vec，Glove 等等)，每一种 embedding 上都进行了 power-means 操作。把这些来自不同 embedding spaces 的 $s^{(i)}$连接起来，叫做 Concatenated p-mean Word Embeddings，作为一个多维度向量，作为 Logistic Regression 的 Input。

Model	Σ	AM	AC	CLS	MR	CR	SUBJ	MPQA	SST	TREC
Arithmetic mean										
GloVe (GV)	77.2	50.0	70.3	76.6	77.1	78.3	91.3	87.9	80.2	83.4
GoogleNews (GN)	76.1	50.6	69.4	75.2	76.3	74.6	89.7	88.2	79.9	81.0
Morph Specialized (MS)	73.5	47.1	64.6	74.1	73.0	73.1	86.9	88.8	78.3	76.0
Attract-Repel (AR)	74.1	50.3	63.8	75.3	73.7	72.4	88.0	89.1	78.3	76.0
GV ⊕ GN	78.3	52.8	71.0	76.8	77.9	78.6	91.6	88.6	81.5	86.2
GV ⊕ GN ⊕ MS	78.7	53.5	70.9	77.0	77.9	79.6	**91.9**	88.9	81.6	86.6
GV ⊕ GN ⊕ MS ⊕ AR	**79.1**	**53.9**	**71.1**	**77.2**	**78.2**	**79.8**	91.8	**89.1**	**82.8**	**87.6**
***p*-mean** [p-values]										
GV $[-\infty, 1, \infty]$	77.9	54.4	69.5	76.4	76.9	78.6	92.1	87.4	80.3	85.6
GN $[-\infty, 1, \infty]$	77.9	55.6	71.4	75.8	76.4	78.0	90.4	88.4	80.0	85.2
MS $[-\infty, 1, \infty]$	75.8	52.1	66.6	73.9	73.1	75.8	89.7	87.1	79.1	84.8
AR $[-\infty, 1, \infty]$	77.6	55.6	68.2	75.1	74.7	77.5	89.5	88.2	80.3	89.6
GV ⊕ GN ⊕ MS ⊕ AR $[-\infty, 1, \infty]$	**80.1**	**58.4**	**71.5**	**77.0**	**78.4**	**80.4**	**93.1**	**88.9**	**83.0**	**90.6**
Baselines										
GloVe + SIF	76.1	45.6	72.2	75.4	77.3	78.6	90.5	87.0	80.7	78.0
Siamese-CBOW	60.7	42.6	45.1	66.4	61.8	63.8	75.8	71.7	61.9	56.8
Sent2Vec	76.8	52.4	**72.7**	75.9	76.3	80.3	91.1	86.6	77.7	78.2
InferSent	**81.7**	**60.9**	72.4	**78.0**	**81.2**	**86.7**	**92.6**	**90.6**	**85.0**	**88.2**

Table 2: Monolingual results. Brackets show the different p-means that were applied to all individual embeddings.

图中每一列代表不同的 NLP 的任务。为了和 InferSent 这样的经典工作对标，这些任务和 InferSent 的实验基本上一致的。第一列是各项任务的加权平均。实验中用到的算法主要分为三大类: 最简单的算术平均(即 p=1)， p−mean(最大，最小，及平均)，过去的经典算法(比如前文介绍过的 SIF，InferSent 等等)。

结论：

1.在第一类算法内部，很显然不同 embeddings 放在一起要好于单 embedding，这个和过去的很多研究是一致的。有些深度学习的算法同时使用不同的 embeddings 并和 CV 类比的称为 channels。显然不同的 embeddings 有互补的信息。

2.比较第一和第二类算法，显然不同的 p 值引入了更丰富的信息。直观上讲最大和最小值的确能提供最显著(数值上)的特征，这和过去的经验相符。当然，把 1 和 2 合并起来使用，则包含不同 p 值的不同 embeddings 串在一起时效果最好。

3.作为第三类算法，只有 InferSent 效果要略好于作者的算法。特别是其他的第三类算法综合表现都逊于第一类算法中的 GloVe。

分析：

1.为什么把 p−means 连接起来有用？

显然不同的 p 值会提供不同的信息。平均值能够提供一些信息。但是不同的句子可以给出一样的平均值，这时给定 max 和 min 的值就能够限定句子里面 embeddings 的取值范围，减少了不确定性。

2.哪些 p 值会更有效？

大的|p|会很快的收敛到 min(p=−∞)和 max(p=∞)。所以除了取最大和最小值之外，能够提供有用信息的 p 值一般较小。当 p 是整数时，奇数比偶数要更有用，因为偶数 p 值会丢失符号信息。另外 p 为正数时比负数时更好(比如以上实验中 p=−1 带来了负效果)。

论文利用了不同的 embeddings 的信息的互补。之前的研究常建议多个 embeddings 混用，这就是一个成功的例子。这些互补信息的 embeddings 直联以后增大了输入的维度，更加方便 logistic regression。 第二是利用了不同的简单操作。比如取平均是简单操作，取最大和最小值都是简单操作，多个简单操作 concatenation 就是复杂操作。这个复杂操作的结果就是对句子整体信息的有效压缩，可以看到这种做法是有效的。

最后，产生句子 embedding 的方法未必就非要经过 LSTM 和 CNN 之类的深度学习模型，可以多做尝试选择合适的模型。但中文语料中效果好坏就看各位大佬的了=。=