# 文章标题

# Word Embeddings with Limited Memory

## 文章总结：

**研究对象**：词向量系统

**实验条件**：有限内存，通过内存的限制来评估词向量的系统，想进一步的优化占用内存的问题时，需要用有限精度数据的表示和计算两个方面来进行对词向量系统的影响。

**最后结论**：通过8位比特作为词向量的固定比特位数训练数据，不会失去词/短语之间的相似性和解析它们时语句存在的依赖性。

**提出的问题**：能否显著的减少词向量的内存需求（文中一直强调使用和训练的词向量）

**问题解释：** 因为词向量的每个维度的数据是需要用bits来表示的，文章研究的是使用比标准的64位更少的比特数来表示每个维度的数据，并且保证词/句相似性和依赖性保持不变。

**实验结果：**

在所有情况下，在没有性能损失的情况下，如果当前标准为64时，可以使用8位，在某些情况下，每个维度 使用4位也是足够的，从而将所需的空间量减少了16倍。

## 文中综述的国内外情况（引用分析别人的观点，类型分类）

1. 考虑传统的簇编码字

参考的研究是1992年Brown clusters，主要是研究将字簇表示成分层，根据路径编码

2. 词向量使用连续向量来表示离散聚类的思想，可以提高连续词句的预测和泛化能力，我觉得是替代语义的意思。

参考的研究是2003年Bengio的研究，2013a Mikolov，2013b Mikolov的研究

3. Brown集群作为离散特征比连续词向量作为作为特征为命名实体来识别任务更好

参考的是2009年Ratinov和Roth的研究

4. 尝试使用针对每个维度调整的阈值二值化 嵌入，基本上使用小于2个位来表示每个维度，结果是当用作命名实体识别任务的特征，二值化的效果可以达到原始词向量的效果甚至更好。

参考的是2014年Guo的研究

5.embedding vectors施加稀疏约束 可以进一步提高单词相似性和文本分类标准数据集上的表示和性能。

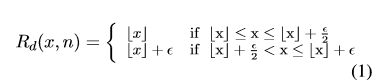
参考的是2015年Faruqui等人的研究

对上面的工作总结得出，对于一些任务，确实不需要所有信息编码在“标准”词向量中，但是二值化会丢失信息，这就需要系统的比较对不同的任务，需要多少比特的词向量才能保持所需的表达性。

## 使用的方法

Value Truncation值截断--将词向量中的任何值x截断为n位表示

1. 使用Post-processing Rounding方法进行四舍五入，保留n位精度。

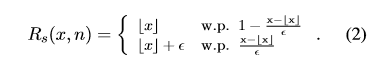


2. 当训练词向量的存储器有限时，我们需要引入新的数据结构，来减少用于对值进行编码的比特位数，来修改训练算法。

参考研究内容是 随机梯度下降（SGD）迭代中使用word2vec算法（Mikolov等人，2013a; Mikolov等人，2013b）

### 有限内存训练

1. 使用随机舍入（Gupta等人，2015）来训练词向量



在这种情况下，即使更新值不足以更新词向量，我们随机地选择一些被更新的值与更新值与舍入精度成比例接近的值。

2. 辅助更新向量

除了直接对值应用舍入的方法之外，我们还提供使用辅助更新向量来换取精度以获得更多空间的方法。假设我们知道SGD中的更新值的范围为[-r0，r0]，并且我们使用附加的m位来存储小于有限数值精度的所有值。这里r0可以很容易地通过运行SGD几个例子估计。

## 实验及输出的结果（可忽略不看）

1. 在维基百科转储数据上使用负样本运行CBOW和skipgram两个模型，并将上下文的窗口大小设置为5。 然后用4比特，6比特和8比特执行值截断。 结果如图1所示，平均结果的数字如表1所示。我们还使用二值化算法（Guo et al。，2014）将每个维度截断为三个值; 这些实验使用图中的“二进制”表示。 对于CBOW和skipgram模型，我们分别训练25和200维的向量。

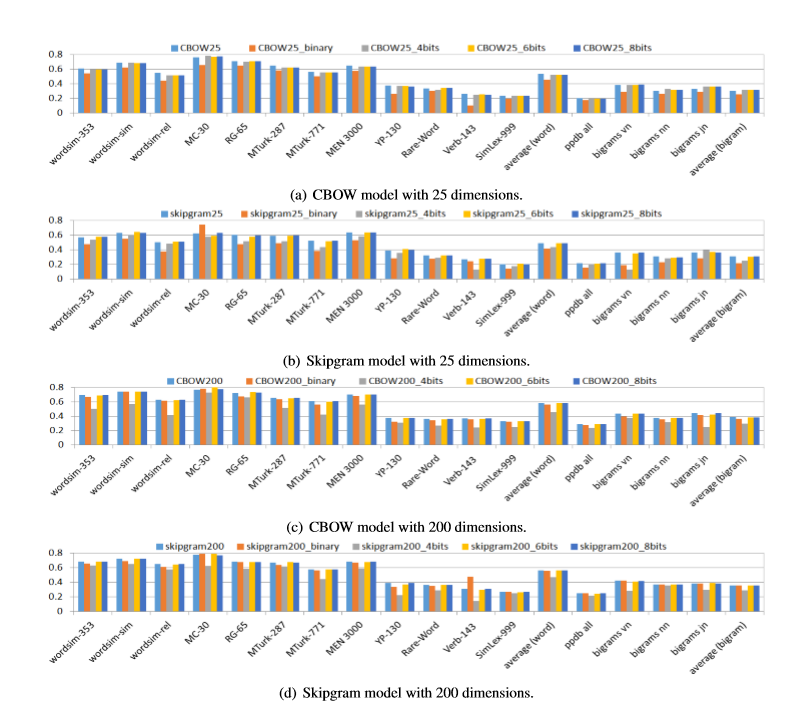


图1 比较具有不同截断值的多个相似性任务的性能。

y轴表示词相似性数据集的斯皮尔曼秩相关系数，以及释义（bigram）数据集的余弦值

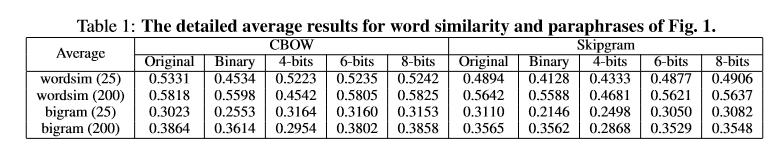


表1 单词相似性和释义的详细平均结果

**结果分析：**

第一个观察是，一般来说，CBOW执行比skipgram模型更好。 当使用截断方法时，存储嵌入所需的存储器显着减少，而测试数据集上的性能保持几乎相同，直到我们截断到4位。 当比较CBOW和skipgram模型时，我们再次看到，对于skipgram模型，4位值的性能下降大于CBOW模型的性能下降。 对于CBOW模型，使用200维时的4位值的性能下降比使用25维时的性能下降更大。 但是，使用跳跃图时，使用25个尺寸时，此下降稍大于200。

我们还评估了二值化方法（Guo et al，2014）。

此模型使用三个值，使用两个位表示。我们观察到，当维度为25时，二进制化比截断更糟。一个可能的解释仅仅是空间的大小;虽然325比字空间的大小大得多，但它不能提供足够的冗余来利用任务中的相似性。因此，二值化方法导致更差的性能。但是，当维度为200时，此方法工作得更好，并且优于4位截断。特别地，二进制化对于skipgram比对于具有200维的CBOW工作更好。一种可能的解释是二进制化算法为单词向量的每个维度计算正负值的平均值，并且使用它来分割该维度中的原始值，因此表现得像将每个维度中的值聚类的模型。二进制化的成功然后表明，跳跃嵌入可能比CBOW嵌入更有区别。

2. 比较表2中的CBOW模型的训练方法。对于随机舍入，我们缩放向上舍入的概率，以确保小梯度值仍然更新值。

随机舍入和具有辅助更新向量的截断（在3.2节中示出）在训练阶段中对于每个值需要16个比特。 具有辅助更新向量的截断最终产生基于8位值的向量，而随机舍入产生基于16位值的向量。 即使我们的辅助更新算法使用较小的内存/磁盘来存储向量，其性能仍然优于随机舍入。 这是因为SGD中的更新值太小而不允许随机舍入方法收敛。 辅助更新向量实现与原始向量非常相似的结果，并且实际上导致与原始截断方法产生的向量几乎相同的向量。

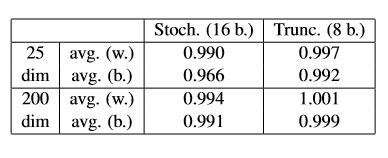


表2 比较训练CBOW模型：我们将原始word2vec嵌入的平均值设置为1，并且表中的值相对于原始嵌入基线。

“avg. （w.）“表示所有词相似性数据集的平均值。

“avg. （b.）“表示所有双字母相似度数据集的平均值。

“Stoch. （16b.）“表示使用应用于16位精度的随机舍入的方法。

“Trunc.（8b.）“表示使用应用于8位精度的8位辅助更新向量的截断的方法。

3. 依赖性实验

我们遵循（Guo等人，2015）在单语环境中的设置。 我们使用不同的截断设置为word2vec词向量训练解析器5000次迭代。

数据集：是CoNLL-X共享任务中的英语数据（Buchholz和Marsi，2006）。

方法：遵循（Guo等人，2015）在使用标记的附着分数（LAS）来评估不同的解析结果。

结果：这里我们只显示200维的词嵌入结果，因为根据经验，我们发现25维结果不如200维稳定。

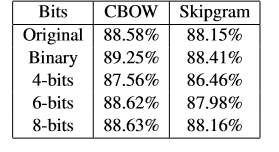


表3 依赖性解析的评估结果（inLAS）。

表3中显示的依赖性解析的结果与单词相似性和释义一致。 首先，我们看到CBOW和skipgram的二值化再次比截断方法更好。 第二，对于截断结果，更多的位导致更好的结果。 使用8位，我们可以再次获得类似于从原始word2vec嵌入获得的结果。

## 使用的数据集

使用多个测试数据集：

1. 词向量相似性（Word Similarity）

Faruqui和Dyer（Faruqui和Dyer，2014）总结的数据集:

wordsim-353，wordsim-sim，wordsim-rel，MC-30，RG-65，MTurk-287，MTurk-771，MEN3000，YP130，Rare -Word，Verb-143和SimLex-999。

参考的研究是1995年Myers,Well研究的，计算词对之间的相似性，并检查计算机和人类标记排名之间的的Spearman相关系数

2. 释义（双字母组或称二元语法）Paraphrases (bigrams)

使用的数据集是(Wieting et al.,2015), ppdb all, bigrams vn, bigrams nn, and bigrams jnn,

用于测试截断是否影响语句级的向量

参考的Wieting等人在2015年使用余弦相似性来评估计算的相似性和注释之间的相似性

## 结论

我们系统地评估词向量的表示大小，对使用它们的NLP任务的性能之前影响有多小。我们的研究考虑了CBOW和skipgram模型分别在25和200维度下的情况，并且表明每个维度用8位（有时甚至更少）足以表示每个值并且可以保持语句任务的性能

我们还提供了两种方法来减少内存使用的嵌入。

## 小编总结：

1. 文章主要是对词向量在有限内存下节省每个维度的比特表示，并且还能够保持表示和语义的依赖，最后得出8位是可以达到最优的。

2. 实验过程对CBOW，Skip-gram模型进行研究，自己google了一下，这个是词向量算法中比较著名的模型。

3. 虽然已经有了一些基于神经网络的语言模型来学习词向量, 但是训练时间都很长, 所以Mikilov等人就直接把隐藏层去掉, 使用了这俩个模型，大大减少了计算量, 提高了计算速度, 然后用更多的数据来训练模型, 最后的效果也不错。