CS5740 Project 4
Word Clusters and Word Vectors

Pengcheng Zhang pz84 | **Jiangjie Man** jm2559 | **Puran Zhang** pz75 | **Jing Liu** jl3556

1 Introduction

One of the tricky problem we faced in Project 1 is how to deal with the sparsity problem in the language model. Compared to the huge and uncountable real human-language, any training corpus is limited, and many word sequences that should not have zero probability will be estimated as zero probability.

In Project 1, we deployed smoothing to mitigate the sparsity problem. In this project, we incorporated word cluster and vectors to handle this issue.

2 Basic observation and analysis

We obtained a pre-trained word cluster resource [1] online and converted the data formats into our 1000 and 3200 word clusters for over 28000 word entries. We included 10 cluster examples (5 from 1000-cluster group and 5 from 3200-cluster group) in Table 1.

For both cluster groups (1000-sized and 3200-sized), we can see that most of the words in each cluster belong to the same syntactic class or words related to the same topic. For example, `1000_a` are mainly preposition, `1000_b` are nouns, `1000_c` are numeral; The 3200 groups listed here are mainly nouns.

Besides, there are also some semantic similarities between words in a cluster. The majority of the words in cluster `1000_a` are prepositions describing position and locality; words in `1000_b` are related to body language; `1000_e` are nouns related to research and report. These patterns are more explicitly shown in 3200 groups. In cluster `3200_a`, words are mainly nouns related to animal or human body; Cluster `3200_b` are nouns about some kinds of food; `3200_c` are mainly finance jargon whereas `3200_e` are accounting jargon.

Some of the clusters are kind of controversial. A good example is Cluster `1000_b`, where body parts such as eyebrows and eyes exist together with emotions such as concerns and suspicions. These two "sub-groups" of words can be divided into two clusters because they are referring to different things, but they can also be clustered together because they are related to body language (the behavior of body parts represents certain emotions).

However, some clusters are really difficult to be understood and explained. The cluster `1000_d` contains words related to time (avail, longer), extent (farther, further), and disability (limp, staggered). It is not clear why these three semantic groups are clustered together. Also, in cluster `3200_d`, the word "manure", which is the feces of animals, are ridiculously clustered together with food and ingredient. It is also not clear to us why Cluster `3200_b` and Cluster `3200_d` are divided even if they are all closely related to food.

3 The Sparsity Problem

The n-gram model we used in Project 1 estimates the probability of the grams using the maximum likelihood estimation (MLE). The MLE has a major problem of sparse data because it is based on only a particular set of training data. For the n-grams that occurred a sufficient number of times, we might be able to have a good estimate of their probabilities. However, since any training corpus is limited by its size compared to the whole language system (say all language available in the world), there will always be a great amount of well accepted English word sequences missing. Thus, for the n-grams that have a small count or even are unknown in the training set, the MLE method produces poor

[1] https://github.com/rug-compling/dep-brown-data

Table 1: Pre-trained cluster examples

Cluster	Words
1000 (cluster)_a	onto, on, between, against, among, outside, overlooking, near ...
1000_b	concerns, eyebrows, questions, toes, doubts, jitters, eyes, suspicions ...
1000_c	half, one-eighth, 1/100th, three-fourths, one-tenth, one-fourth ...
1000_d	avail, longer, doubt, limp, staggered, farther, further ...
1000_e	report, chart, poll, study, survey ...
3200 (cluster)_a	throats, necks, mouths, noses, wounds, hearts ...
3200_b	carrots, pizzas, menus, chocolates, cakes, sandwiches ...
3200_c	profits, charge-offs, write-downs, write-offs, losses, taxes, wages ...
3200_d	lobster, manure, spaghetti, dough, nut, ham, caviar, cane, tuna ...
3200_e	accounting, auditing, cpa, brokerage, insurance, banking ...

estimates of their probabilities.

One may argue that using larger training corpora can solve the data sparsity problem. Yet this solution cannot always work, as large-size training corpora are not always available, and even if they are available, Zipf's Law shows that a large portion of words have only very small frequencies. Thus, the increase of training corpora size may lead to even more infrequent appearance of certain word sequences.

In our original approach in Project 1, we used Good-Turing smoothing to get better estimates for the zero or low-frequency counts in the training corpora. Here in Project 4, we will use word clustering approach in probability calculation to solve the data sparsity problem. The intuition of using word clustering in this process comes from the fact that all words in the same word cluster are similar or related to some extent. Thus, when we want to find out the probability of a word sequence, e.g. bigram $P(w_2|w_1)$, we can look for the probability of their cluster sequence, which is $P(c_2|c_1)$ in this case, and then multiply it by the probability of word appearing in this cluster, which is $P(w_2|c_2)$.

4 Improvement Compared to Project 1

In this section, we compared our newly-developed model with word clustering and original language model developed in Project 1.

4.1 Justification of using Word Cluster

For any given assignment of a word w_i to a cluster (also called a class) c_i, there may be many to many mappings, i.e. a word w_i may belong to more than one cluster, and a cluster c_i will typically contain more than one word. For the sake of simplicity in our project, we assume that a word w_i can only be uniquely mapped to its own cluster c_i.

The cluster-based n-gram model is similar to the word-based n-gram model in that it also make the Markov assumption that a word cluster depends only on the previous word cluster. In the basic cluster-based n-gram model, the conditional probability of a word w_i is defined as the product of the two factors: the probability of the cluster given the preceding clusters, and the probability of a particular word given the cluster. Thus, in cluster-based bigram model that we used in our approach, we have:

$$P(w_i|w_{i-1}) = P(w_i|c_i) * P(c_i|c_{i-1})$$

Based on maximum likelihood estimation (MLE), the probability of the cluster given the previous cluster and the probability of word given the cluster are computed as follows:

$$P(c_i|c_{i-1}) = \frac{C(c_{i-1}c_i)}{C(c_i)}$$

$$P(w_i|c_i) = \frac{C(w_i)}{C(c_i)}$$

Since one cluster may contain multiple words, the frequency of cluster sequences will

be much larger than word sequences, and data sparsity problem can be alleviated. As a result, the cluster-based model can produce a more knowledgeable estimate of the probability of word strings.

4.2 Experiments and Analysis

One experiment is using different clustering size. We experimented with training our model using 320 word clusters, 3200 word clusters, and 10000 word clusters, respectively. As shown in the table below, the Kaggle score for the two experiments are 0.528 for 320 word clusters (cluster_320 in the table), 0.544 for 3200 word clusters (cluster_3200 in the table), and 0.576 for 3200 word clusters (cluster_10000 in the table).

From the the slight improvement (3%) of using 3200 word clusters over using 320 word clusters, we can see that more finely divided clusters can lead to better performance. Increasing the number of word clusters to 10000 resulted in another 5% improvement in the Kaggle score compared to 3200 word clusters, which agrees with the above conclusion, too.

It is interesting to note that when increasing the number of word clusters from 320 to 3200, we only get 3% improvment int the Kaggle score by increasing the cluster number by 10 times, however when increasing the cluster number from 3200 to 10000, which is only 3 times, resulted in a 5% improvement in Kaggle score. The different "improvement efficiency" in the two comparisons may suggest that if the sparsity problem can only be significantly alleviated when the clusters are divided finely enough.

Group	Kaggle score
Unsmoothed word	.40800
Smoothed word	.52000
cluster_320	.52800
cluster_3200	.54400
cluster_10000	.57600

4.3 Quantify the Improvement

In the table above we also listed our Kaggle performance of the word-based bigram model we trained in Project 1. Compared to the Kaggle score of 0.52 of our smoothed word-based bigram model, we can see that integrating word clustering approach certainly improves the score of our topics classification task. As we hypothesized, word clustering mitigated the sparsity issues that affected our performance.

Comparing our approach with 10000 clusters with the smoothed word-based bigram model we trained in Project 1, we can conclude that the classification accuracy of the former improved by 10.8% compared to the latter.

5 Train our own word clusters

We've trained our own word clusters by applying k-means algorithm on a google news word2vec model.

5.1 Approach Introduction

At first the original k-means with default parameters are used to obtain word clusters. Unfortunately even on a server with 20 CPU core, the first iteration took more than one hour. We estimated that more than 400 iteration will be needed,thus it's not feasible to finish the task in a reasonable time frame, instead, we've made adjustment as follows.

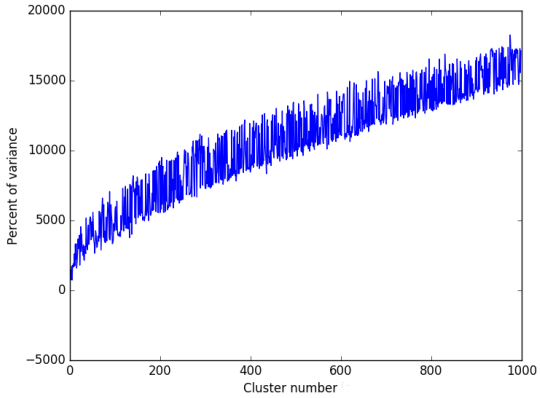
- Instead of the original k-means, Mini Batch K-means is used, it's proved to be efficient on large scale data. The quality of the result is reduced but the difference in quality can be quite small (Sculley, 2010).

- The reassignment ratio, which refers to the ratio size of the smallest cluster and the largest cluster, is set to 0.1 by default. Since the data size is large and there exist many cluster whose size is 1. While the largest cluster(in our case, it's cluster that most words have similar meaning) can be easily exceed 0.1, we set the reassignment ratio to 0.001.

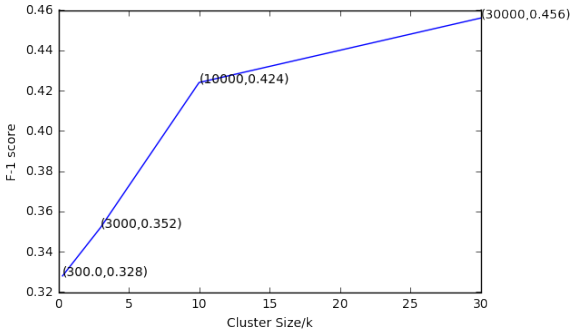
5.2 Selecting cluster number k

A frequent problem in dividing data set using k-means is determining the cluster number k.

In this project, we used the Elbow method on a randomly sampled google news word2vec model and the results has been shown in the photo below. We haven't used some hierarchical clustering or other clustering method such as x-means clustering due to the data set is too large. The results has shown that the marginal gain drop at around cluser size = 6,000,000.



Since results from sample size might not be enough, we've also tested our program using different cluster size, and the result is summed below:



It can be seen that F-1 score increase more rapidly when cluster size is smaller than 10000, more slowly when cluster size is smaller than 10000 so we chose cluster size $k = 300000$ as our final one since a larger cluster size may take too long to run.

5.3 Observations and Analysis

Five examples of the word clusters that we trained are shown in Table 2. In Cluster `self_a`, the words are related to food and cooking. Words in Cluster `self_b` are commonly used verbs, and those in Cluster `self_c` are related to Chinese soccer teams. Cluster `self_d` contains food, and Cluster `self_e` contains words related to victory.

An interesting observation is the difference between Cluster `self_a` and Cluster `self_d`. These two word clusters are both related to food, but they are divided into different clusters. By carefully examine the words in both clusters, we find that the Cluster `self_a` are related to raw food and cooking ingredients which are not directly edible, while Cluster `self_d` are related to cooked food or directly edible food.

5.4 Comparison with Pre-trained Clusters

From the semantic aspect, our self-trained clusters have better performance than the pre-trained ones. While it is difficult to understand and explain why words that are of different "sub-groups" are grouped in some of the pre-trained clusters, and why two pre-trained clusters do not have apparent difference from each other, our self-trained clusters have reasonable semantic differences from each other and have clear similarities within the cluster.

However, from the syntactic aspect, the pre-trained clusters have slightly better performance than our self-trained ones. The pre-trained clusters divide words of different parts of speech (i.e. verbs, nouns, preposition, etc) into different clusters, while our self-trained clusters do not complete differentiate between them. For example, our cluster `self_a` puts the adjective "braised" and "pan-seared" together with other nouns (they are closely related in semantics, i.e. in cooking, though), and `self_e` groups verbs "win" and "won" together with nouns like "triumph" and "victories".

Our result has shown that a cluster with larger corpus and larger cluster size will yield a better results. However, it's suggested that euclidean distance might not be adequate for clustering (Béjar Alonso, 2013), more clustering method should be tried in the future.

6 Team Member Contribution

Our work of distribution is listed as follow:

Table 2: Self-trained cluster examples

Cluster	Words
self_a	strip_steak, pan_seared, duck_breast, beef_tenderloin, braised, garlic_sauce ...
self_b	Get, Give, Bring, Taking, Want, Choose, Take, Leave, Took ...
self_c	Zheng_Tao, Shenzhen, Shanghai_Shenhua, Beijing_Guoan, Dalian_Shide ...
self_d	baked_potato, meat_loaf, sausage_gravy, shortcake, banana_cream ...
self_e	winners, triumph, win, clinched, victories, won, defeating ...

- Pengcheng Zhang and Jiangjie Man: Apply the word clutsering/vectorization package to the training set and topic clas-sifciation.

- Puran Zhang and Jing Liu: Observation analysis, problem identification and re-sult analysis.

References

Javier Béjar Alonso. 2013. K-means vs mini batch k-means: a comparison.

David Sculley. 2010. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178. ACM.