Word Clustering for Data Sparsity: A Literature Survey

Kashyap Popat
113050023

June 16, 2013

In this report, we present the literature survey done for our work with SA and other NLP applications. The road map of this report is as follows. In Section-1, we introduce clustering process and describe a few existing word clustering techniques. Section-2 talks about the smoothing process followed by why clustering is better for our work in Section-3. Finally in Section-4, we talk about the related work done for different NLP applications in which word clusters are used as helpful features.

1 Clustering: The Savior

The sparsity problem is a very familiar problem in the regimes of statistical modeling. Clustering is one of the techniques, which can be used to solve the sparsity problem. After introducing the clustering process, in this section, we describe the **Brown Clustering algorithm** (Brown et al., 1992), the **Predictive Exchange algorithm** (Uszkoreit and Brants, 2008) and the **Cross-lingual Clustering algorithm** (Täckström et al., 2012). We use these clustering techniques in our approaches for handling sparsity problem in different NLP applications. In the next subsection, we introduce the word clustering process.

1.1 What is Clustering?

Cluster analysis or **clustering** is a task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar (in some sense or another) to each other than to those in other clusters[1]. It can be considered as the most important unsupervised learning problem. It deals with finding a structure in a collection of unlabelled data[2]. The clustering techniques are very useful in reducing sparsity because of their inherent nature of achieving abstraction. We can represent the clustering process as shown in Figure-1.

In Figure-1, we can easily identify that the data can be divided into three clusters. For deciding whether to keep two objects into one cluster or not, we can use various types of similarity criterion.

[1] `http://en.wikipedia.org/wiki/Cluster_analysis` (accessed 11 June, 2013)
[2] `http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/` (accessed 11 June, 2013)

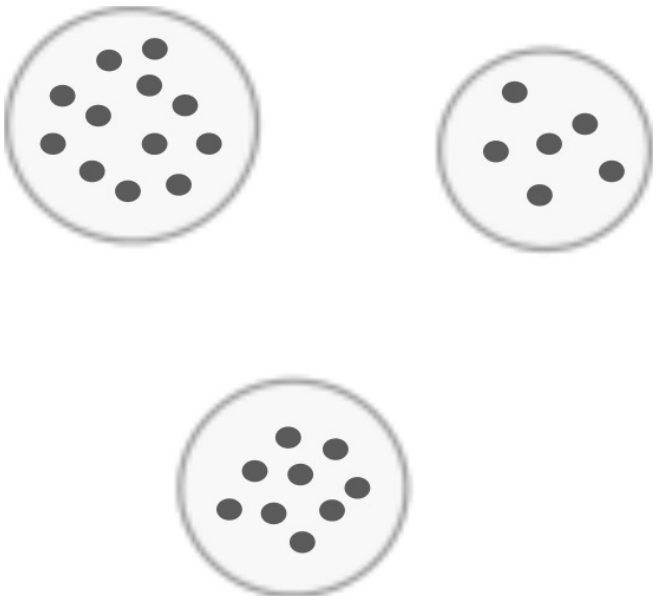

Figure 1: Clustering

This criterion can be distance, *i.e.*, two or more objects belong to the same cluster if they are "close" according to a given distance (in above case it is geometrical distance). This is known as **distance-based clustering**.

Another kind of clustering is **conceptual clustering** in which two or more objects belong to the same cluster if they define a concept common to all other objects in that cluster. In other words, objects are grouped according to their fit to descriptive concepts, not according to simple similarity measures.

A categorization of major clustering algorithms and short description about them are as follows (Han, 2005):

- **Partition methods:** Given a dataset of n objects or data tuples, a partition method constricts k partitions of the data, where each partition represents a cluster and $k \leq n$.
- **Hierarchical methods:** This method creates a hierarchical decomposition of the given set of data objects. Based on how the hierarchical decomposition is formed, it can have two different types: *agglomerative approach (bottom-up)* and *divisive (top-down)*.
- **Density-based methods:** Distance based clustering methods can only find the spherical-shaped clusters and face difficulties in finding clusters of arbitrary shapes. In this method, the general idea is to continue growing the given cluster so long as the density (number of objects or data points) in the neighborhood exceeds some threshold. This approach can find the clusters with arbitrary shapes as well.
- **Grid-based methods:** This method quantizes the object space into finite number of cells which form a grid structure. It then performs all of clustering operations on the grid structure. The main

advantage of this approach is its fast processing time which is typically independent of the number of data objects.

- **Model-based methods:** This method hypothesizes a model for each of the clusters, and finds the best fit of the data to that model.

For clustering of words in the text, the techniques are mainly classified into two types:

- **Word clustering based on paradigmatic analysis:** This is same as conceptual clustering. WordNet is a byproduct of such clustering based on paradigmatic analysis. In WordNet, paradigms are manually generated based on the principles of lexical and semantic relationship among words (Fellbaum, 1998).

- **Word clustering based on syntagmatic analysis:** This type of clustering techniques concentrate on the surface properties of the text. Brown clustering algorithm, predictive exchange algorithm, and corss-lingual clustering fall into this category.

In the next subsection, we present a few applications of the clustering in general apart from solving data sparsity problem.

1.2 Applications of Clustering

The goal of clustering is to determine the intrinsic grouping in a set of unlabelled data. For deciding the quality of the clustering, there is no absolute best criterion as such which would be independent of the final aim of the clustering. Consequently, it is the user which must supply this criterion, in such a way that the result of the clustering will suit their needs. For instance, we could be interested in finding representatives for homogeneous groups (data reduction), in finding natural clusters and describe their unknown properties, in finding useful and suitable groupings, in finding unusual data objects (outlier detection) *etc.* Some of the possible applications of clustering are as following[3]:

- **Marketing:** finding groups of customers with similar behavior given a large database of customer data containing their properties and past buying records

- **Biology:** classification of plants and animals given their features

- **Libraries:** book ordering

- **Insurance:** identifying groups of motor insurance policy holders with a high average claim cost; identifying frauds

- **City-planning:** identifying groups of houses according to their house type, value and geographical location

- **Earthquake studies:** clustering observed earthquake epicenters to identify dangerous zones

[3] `http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/` (accessed 11 June, 2013)

- **WWW:** document classification; clustering web-log data to discover groups of similar access patterns

In the next subsection, we see how clustering is helpful in addressing data sparsity problem in different NLP applications.

1.3 How Clustering Reduces Sparsity Problems

As explained earlier, **data sparsity problem** occurs in the setting of supervised statistical learning method, when some data from the test side is not present in the training dataset. The size of the model becomes large when the number of features used in modeling process is very large. In addition, large number of features make the feature vectors sparse and pull down the accuracy of the statistical model. We describe the effects of clustering on above points one by one.

Effects on data sparsity problem

In case of a data in lexeme space, it is very likely that some of the test data may not be present in the training data. Since the clustering process allows us to transfer the data from lexeme space to cluster space, we can reduce this problem of data sparsity. While using the data in cluster space, it is very unlikely to come across such data from the test side which is not present in the training data. The reason for that is: number of unique clusters are lesser as compared to the unique number of words. Thus, by transferring data from lexeme space to cluster space with the help of clustering process, we decrease the data sparsity problem up to large extent.

Effects on the size of the model

When we use words as features in the statistical modeling process, most of the times we end up having a large number of features. Typically, for statistical modeling techniques which use surface words as features, the size of feature vectors become same as number of unique words in the data. This makes the statical model very large. When we convert the data in lexeme space to cluster space, we reduce the vocabulary of the data (as the number of clusters are lesser than the number of unique words). This reduces the number of features in the statistical modeling and also reduces the size of the model by dimensionality reduction. This also leads to reduction in noise (Cunningham, 2008). Apart from this, it has also been shown that the simple hypothesis or model performs better than the complex models with large feature set.

The next subsection explains different word clustering methods based on syntagmatic analysis. We use these methods in our approach to reduce sparsity problem in different NLP applications.

1.4 Brown Clustering Algorithm

Brown clustering algorithm (Brown et al., 1992) is a bottom-up agglomerative clustering algorithm which generates hard clustering, *i.e.*, each word belongs to one cluster. The primary motivation of this co-occurrence based algorithm is to learn the class based language model. However, it can also be applied to any statistical problem where there is a need of reducing sparsity. Since this technique is co-occurrence based, it is able to extract the classes that have the flavor of either syntactically based groupings or semantically based grouping, depending on the nature of the underlying statistics.

1.4.1 Methodology

It is a well known fact that some words are similar to other word in their meaning and syntactic function. For example, it would not be surprising to learn that the probability distribution of words in the neighborhood of the word *Thursday* is very much like that for words in the neighborhood of the word *Friday*. Based on this assumption, the algorithm tries to cluster surface words based on co-occurrence probabilities.

This hierarchical word clustering algorithm takes a large corpus of sentences as an input and gives a partition of the words into clusters as an output. The mathematical formulation of the same is as follows [4]:

Let us assume that,

- V is the vocabulary of the corpus
- $\mathbf{w} : w_1, w_2, ..., w_m$ is the word sequence with $w \in V \cup S$, where S is the special starting symbol
- $n(w, v)$ is the number of times w precedes v in the corpus
- $n(w)$ is the number of times w has occurred in the corpus

If, $\mathcal{C} : V \rightarrow 1, 2, ...k$ is a partition function of the vocabulary into k classes, the likelihood model is defined as:

$$P(\mathbf{w}; \mathcal{C}) = \prod_{i=1}^{m} p(w_i|\mathcal{C}(w_i)) \cdot p(\mathcal{C}(w_i)|\mathcal{C}(w_{i-1})) \tag{1}$$

In above equation, the bi-gram probabilities are used because it has been observed that by using bi-gram probabilities, the maximum likelihood assignment of words to classes is equivalent to the assignment for which the average mutual information of adjacent classes is greatest. Above equation can be written down in a more convenient way as:

$$\log P(\mathbf{w}; \mathcal{C}) = \sum_{i=1}^{m} \log[p(w_i|\mathcal{C}(w_i)) \cdot p(\mathcal{C}(w_i)|\mathcal{C}(w_{i-1}))] \tag{2}$$

To define the quality of a clustering, Liang (2005) view the clustering process in the context of a class-based bigram language model as shown in Figure-2. Given a clustering $\mathcal{C}$ that maps each word to

[4] `http://www.cs.columbia.edu/~mcollins/courses/e6998-3/lectures/lec11.pdf` (accessed 10 October, 2012)

a cluster, the class-based language model assigns a probability to the input text $w_1, ..., w_m$, where the maximum-likelihood estimate of the model parameters (estimated with empirical counts) are used. The quality of the clustering $\mathcal{C}$ is defined as the logarithm of this probability (Equation-2) normalized by the length of the text:

$$
\begin{aligned}
Quality(\mathcal{C}) &= \frac{1}{m}\sum_{i=1}^{m}\log p(w_i|\mathcal{C}(w_i)) \cdot p(\mathcal{C}(w_i)|\mathcal{C}(w_{i-1})) \\
&= \sum_{w,w'}\frac{n(w,w')}{m}\log p(\mathcal{C}(w')|\mathcal{C}(w)) \cdot p(w'|\mathcal{C}(w')) \\
&= \sum_{w,w'}\frac{n(w,w')}{m}\log\frac{n(\mathcal{C}(w),\mathcal{C}(w'))}{n(\mathcal{C}(w))} \cdot \frac{n(w')}{n(\mathcal{C}(w'))} \\
&= \sum_{w,w'}\frac{n(w,w')}{m}\log\frac{n(\mathcal{C}(w),\mathcal{C}(w')) \cdot m}{n(\mathcal{C}(w)) \cdot n(\mathcal{C}(w'))} + \sum_{w,w'}\frac{n(w,w')}{m}\log\frac{n(w')}{m} \\
&= \sum_{c,c'}\frac{n(c,c')}{m}\log\frac{n(c,c') \cdot m}{n(c) \cdot n(c')} + \sum_{w'}\frac{n(w')}{m}\log\frac{n(w')}{m} \\
&= \sum_{c,c'}p(c,c')\log\frac{p(c,c')}{p(c) \cdot p(c')} + \sum_{w}p(w) \cdot \log p(w) \\
&= I(\mathcal{C}) - H
\end{aligned}
\tag{3}
$$

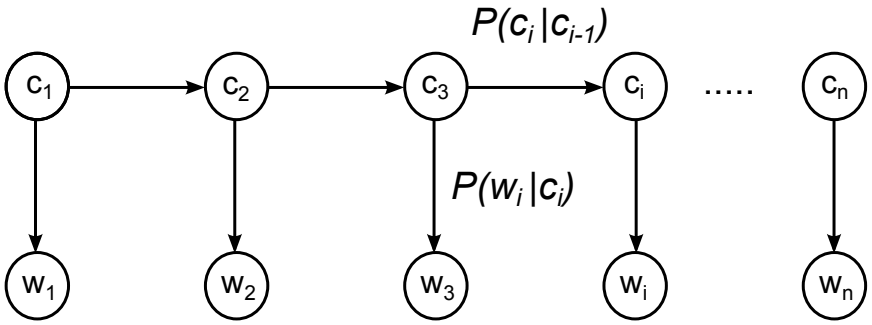

Figure 2: The class-based bigram language model: quality of a clustering

Where, $p(c,c') = \frac{n(c,c')}{m}, p(w) = \frac{n(w)}{m}$ and $p(c) = \frac{n(c)}{m}$. The first term $I(\mathcal{C})$ is the mutual information between adjacent clusters and the second term H is the entropy of the word distribution. Note that the quality of $\mathcal{C}$ can be computed as a sum of mutual information weights between clusters minus the constant H, which does not depend on $\mathcal{C}$. This decomposition allows to make optimizations. There were improvements in the brown clustering algorithm based on optimization techniques. The different versions of this algorithm are described as following.

First algorithm

In this algorithm, the aim is to find k clusters. The algorithm is defined as follows:

Step-1: Start with $|V|$ clusters: each word gets its own cluster

Step-2: Run $|V| - k$ merge steps:

- At each merge step, pick two clusters c_i and c_k, and merge them into a single cluster such that, $Quality(\mathcal{C})$ for the clustering $\mathcal{C}$ after the merge step is maximized at each stage

This naive algorithm has complexity of $O(|V|^5)$. Even after the improvements by Brown et al. (1992), it still remains $O(|V|^3)$.

Second algorithm

Step-1: Take the top k most frequent words and put them into its own cluster: $c_1, c_2, ..., c_k$

Step-2: For $i = (k+1)...|V|$:

- Create a new cluster, c_{k+1}, for i^{th} most frequent word, which will create $k+1$ clusters
- Choose two clusters from $c_1...c_{k+1}$ to be merged such that new clustering gives maximum value for $Quality(\mathcal{C})$, which will again create k clusters

Running time of this algorithm is: $O(|V| \cdot k^2 + n)$, where n is the corpus length and k is the number of clusters.

Sample clusters after the clustering process are given in Figure-3. In the next subsection, we describe the predictive exchange algorithm which is similar to the brown clustering algorithm.

1.5 Predictive Exchange Algorithm

Uszkoreit and Brants (2008) propose the predictive class-based bi-gram model as:

$$P(\mathbf{w}; \mathcal{C}) = \prod_{i=1}^{m} p(w_i|\mathcal{C}(w_i)) \cdot p(\mathcal{C}(w_i)|w_{i-1}) \tag{4}$$

This model is again a co-occurrence based model like brown cluster algorithm. Compare this to the model of brown clustering algorithm:

$$P(\mathbf{w}; \mathcal{C}) = \prod_{i=1}^{m} p(w_i|\mathcal{C}(w_i)) \cdot p(\mathcal{C}(w_i)|\mathcal{C}(w_{i-1}))$$

Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends Sundays Saturdays
June March July April January December October November September August
people guys folks fellows CEOs chaps doubters commies unfortunates blokes
down backwards ashore sideways southward northward overboard aloft downwards adrift
water gas coal liquid acid sand carbon steam shale iron
great big vast sudden mere sheer gigantic lifelong scant colossal
man woman boy girl lawyer doctor guy farmer teacher citizen
American Indian European Japanese German African Catholic Israeli Italian Arab
pressure temperature permeability density porosity stress velocity viscosity gravity tension
mother wife father son husband brother daughter sister boss uncle
machine device controller processor CPU printer spindle subsystem compiler plotter
John George James Bob Robert Paul William Jim David Mike
anyone someone anybody somebody
feet miles pounds degrees inches barrels tons acres meters bytes
director chief professor commissioner commander treasurer founder superintendent dean custodian
liberal conservative parliamentary royal progressive Tory provisional separatist federalist PQ
had hadn't hath would've could've should've must've might've
asking telling wondering instructing informing kidding reminding bothering thanking deposing
that tha theat
head body hands eyes voice arm seat eye hair mouth

Figure 3: Sample Clusters (Brown et al., 1992)

The use of class-to-class transitions can lead to more compact models, which also helps in addressing data sparsity. But Uszkoreit and Brants (2008) propose that reliable statics can directly be obtained on the word-to-class transitions while clustering the large data sets.

In this algorithm, they modify the exchange algorithm (Kneser and Ney, 1993), which is based on the model proposed by Brown et al. (1992). The mathematical formulation of the predictive exchange algorithm is as follows:

$$\begin{aligned} P(\mathbf{w};\mathcal{C}) &= \prod_{i=1}^{m} p(w_i|\mathcal{C}(w_i)) \cdot p(\mathcal{C}(w_i)|w_{i-1}) \\ &= \prod_{i=1}^{m} \frac{N(w_i)}{N(\mathcal{C}(w_i))} \cdot \frac{N(w_{i-1},\mathcal{C}(w_i))}{N(w_{i-1})} \end{aligned} \tag{5}$$

Where, $N(w)$ denotes the number of the word w in the training corpus and $N(v,c)$ the number of occurrences of the word v followed by some word in class c. Then, the following optimization criterion can be derived, with $LL(\mathcal{C})$ being the log likelihood function of the predictive class bigram model given

Algorithm 1 Predictive Exchange Algorithm

Input: The fixed number of cluster N_c

Output: Compute initial clustering

```
while clustering changed in last iteration do
    for w ∈ V do
        for c ∈ C do
            move word w tentatively to cluster c
            compute updated optimization criterion
        end for
        move word w to cluster maximizing optimization criterion
    end for
end while
```

a clusterng $\mathcal{C}$:

$$\begin{aligned} LL(C) &= \sum_{w \in V} N(w) \cdot \log p(w|\mathcal{C}(w)) + \sum_{v \in V, c \in \mathcal{C}} N(v,c) \cdot \log p(c|v) \\ &= \sum_{w \in V} N(w) \cdot \log \frac{N(w)}{N(C(w))} + \sum_{v \in V, c \in \mathcal{C}} N(v,c) \cdot \log \frac{N(v,c)}{N(v)} \\ &= \sum_{w \in V} N(w) \cdot \log N(w) - \sum_{w \in V} N(w) \cdot \log N(C(w)) \\ &\quad + \sum_{v \in V, c \in \mathcal{C}} N(v,c) \cdot \log N(v,c) - \sum_{v \in V, c \in \mathcal{C}} N(v,c) \cdot \log N(v) \end{aligned} \tag{6}$$

The last summation of Equation-6 effectively sums over all occurrences of all words and thus cancels out with the first summation of Equation-6 which leads to:

$$LL(\mathcal{C}) = \sum_{v \in V, c \in \mathcal{C}} N(v,c) \cdot \log N(v,c) - \sum_{w \in V} N(w) \cdot \log N(\mathcal{C}(w)) \tag{7}$$

In the first summation of Equation-7, for a given word v only the set of classes which contain at least one word w for which $N(v,w) > 0$ must be considered, denoted by $suc(v)$. The second summation is equivalent to $\sum_{c \in \mathcal{C}} N(c) \cdot \log N(c)$. This will simplify the criterion:

$$LL(\mathcal{C}) = \sum_{v \in V, c \in suc(v)} N(v,c) \cdot \log N(v,c) - \sum_{c \in \mathcal{C}} N(c) \cdot \log N(c) \tag{8}$$

Based on above mathematical formulation, the predictive exchange algorithm is as shown in Algorithm-1 (Uszkoreit and Brants, 2008). For efficiency reasons, an exchange of a word between two clusters is separated into a *remove* and a *move* procedure. In each iteration the *remove* procedure only has to be called once for each word, while for a given word *move* is called once for every cluster to compute the consequences of the tentative exchanges. The *move* method is given in Algorithm-2 (Uszkoreit and Brants, 2008). The *remove* method is similar.

In the next subsection, we talk about one more clustering algorithm, the cross-lingual clustering proposed by Täckström et al. (2012).

Algorithm 2 Procedure *move*

Input: A word w, and a destination c

Output: The change in the optimization criterion when moving w to c

```
delta ← N(c) · log N(c)
N'(c) ← N(c) − N(w)
delta ← delta − N'(c) · log N'(c)
if not a tentative move then
    N(c) ← N'(c)
end if
for v ∈ suc(w) do
    delta ← delta − N(v, c) · log N(v, c)
    N'(v, c) ← N(v, c) − N(v, w)
    delta ← delta + N'(v, c) · log N'(v, c)
    if not a tentative move then
        N(v, c) ← N'(v, c)
    end if
end for
```

1.6 Cross-lingual Clustering Algorithm

In any cross-lingual application, the essential requirement is to bridge the language gap by transferring linguistic structure from one language to other. To achieve this, one of the obvious ways is to use the surface level words and transfer them to other language using an Machine Translation (MT) system. Use of an MT system for such cross-lingual applications is not motivated because it requires that the respective language pair has an efficient MT system.

Based on this idea, Täckström et al. (2012) proposed a cross-lingual technique which uses word clusters to bridge the language gap in the cross-lingual applications. The methodology of the algorithm is explained in the next subsection.

1.6.1 Methodology

Naive method of generating cross-lingual clustering has two stages. In the first stage, it clusters the source language (S) using modified exchange algorithm (Uszkoreit and Brants, 2008) and then using word alignments it projects these clusters to a target language (T). Given two aligned word sequences $w^S = \langle w_i^S \rangle_{i=1}^{m_S}$ and $w^T = \langle w_i^T \rangle_{j=1}^{m_T}$, let $\alpha^{T|S}$ be a set of scored alignments from source language to the target language, where $(w_j^T, w_{a_j}^S, s_{j,a_j}) \in \alpha^{T|S}$ is an alignment from the ${a_j}^{\text{th}}$ source word to j^{th} target word, with score $s_{j,a_j} \geq \delta$. The shorthand $j \in \alpha^{T|S}$ is used to denote those target words w_j^T that are aligned to some source word $w_{a_j}^S$. Given the source side clustering $\mathcal{C}^S$, they align the target word $t \in \mathcal{C}^T$

to the cluster with which it is most often aligned:

$$LC(t) = \underset{k}{\text{argmax}} \sum_{\substack{j \in \alpha^{T|S} \\ s.t. \mathcal{C}^T(w_j^T)=t}} s_{j,a_j} \left[\mathcal{C}^S(w_{a_j}^S) = k \right] \tag{9}$$

Where, $[\cdot]$ is the indicator function.

As per Täckström et al. (2012), there are two potential drawbacks with this naive approach:

- Since the approach uses word alignments, it only provides a clustering of the target words which occur in the word aligned data. The word aligned data is typically smaller than the monolingual data sets.
- The projected clustering on the target side may not necessarily correspond to an acceptable clustering in terms of monolingual likelihood.

To resolve these issues, a complex model is proposed, which tries to find the clustering which is good according to both source and target language. The intuition of this approach is that the clusterings $\mathcal{C}^S$ and $\mathcal{C}^T$ are forced to jointly explain the source and target data, treating the word alignments as a form of soft constraints.

They use the model of predictive exchange algorithm to cluster the large monolingual data sets. These likelihood functions are denoted as $L^S(w^S; \mathcal{C}^S)$ and $L^T(w^T; \mathcal{C}^T)$ for source and target language respectively (as defined in Equation-4). Here, word sequences denoted by w^S and w^T denote large monolingual non-aligned data sets. Afterwards, these monolingual likelihood functions are coupled with additional factors based on word alignments:

$$L^{T|S}(w^T; \alpha^{T|S}; \mathcal{C}^{\mathcal{T}}; \mathcal{C}^S) = \prod_{j \in \alpha^{T|S}} p(w_j^T | \mathcal{C}^T(w_j^T)) \cdot p(\mathcal{C}^T(w_j^T) | \mathcal{C}^S(w_{a_j}^S))$$

and the symmetric $L^{S|T}(w^T; \alpha^{S|T}; \mathcal{C}^{\mathcal{S}}; \mathcal{C}^T)$. By combining all four factors, the joint monolingual and cross-lingual objective function can be defined as:

$$L^{S,T}(w^S, w^T; \alpha^{T|S}, \alpha^{S|T}; \mathcal{C}^{\mathcal{S}}; \mathcal{C}^T) = L^S(...) \cdot L^T(...) \cdot L^{T|S}(...) \cdot L^{S|T}(...) \tag{10}$$

They also proposed an algorithm to approximately optimize Equation-10 with the alternating procedure as given in Algorithm-3. In this algorithm, they iteratively maximize L^S and L^T, keeping the other factors fixed. This technique is referred as **X-Lingual clustering**.

Thus, in this section, we introduced clustering, its applications and how it helps to solve the data sparsity problem. We also described different clustering algorithms which we use to tackle data sparsity problem in various NLP applications. There is one more technique which goes hand in hand with data sparsity problem, *i.e.*, smoothing. In the next section, we introduce smoothing and a few smoothing techniques.

Algorithm 3 Cross-lingual Clustering (XC)

Input: Source and target language corpora, word aligned corpus

Output: Cross-lingual clusters

1: ## $\mathcal{C}^S$, $\mathcal{C}^T$ *randomly initialized*

2: **for** $i \leftarrow 1$ to N **do**

3: Find $\mathcal{C}^S_* \approx \text{argmax}_{\mathcal{C}^S} L^S(w^S; \mathcal{C}^S)$

4: Project $\mathcal{C}^S_*$ to $\mathcal{C}^T$ using Equation-9 by keeping clusters of non-projected words in $\mathcal{C}^\mathcal{T}$ fixed

5: Find $\mathcal{C}^T_* \approx \text{argmax}_{\mathcal{C}^T} L^T(w^T; \mathcal{C}^T)$

6: Project $\mathcal{C}^T_*$ to $\mathcal{C}^S$ using Equation-9 by keeping clusters of non-projected words in $\mathcal{C}^\mathcal{S}$ fixed

7: **end for**

2 Smoothing: An Alternative

In this section, we explain the smoothing techniques, which also addresses the sparsity problem. We explain the smoothing phenomenon from Chen and Goodman (1996), in which they present the smoothing techniques in context of n-gram language models. However, the techniques are generic in terms of their usages. In the next subsection, we introduce a task of smoothing.

2.1 What is Smoothing?

In statistics and image processing, to **smooth** a data set is to create an approximating function that attempts to capture important patterns in the data, while leaving out noise or other fine-scale structures/rapid phenomena [5]. Smoothing may be used in two important ways that can aid in data analysis:

- By being able to extract more information from the data as long as the assumption of smoothing is reasonable
- By being able to provide analysis that are both flexible and robust

The term **smoothing** describes techniques for adjusting the maximum likelihood estimate of probabilities to produce more accurate probabilities. The name *smoothing* comes from the fact that these techniques tend to make distributions more uniform, by adjusting low probabilities such as zero probabilities upward and high probabilities downward. Not only do smoothing techniques generally prevent zero probabilities, but they also attempt to improve the accuracy of the model as a whole. In the next subsection, we explain the need of smoothing in statistical modeling[6].

2.2 Motivation

The main motivation behind using smoothing techniques is to remove the sparsity problem, which is very likely to occur in case of any statistical modeling. Let us say, we have a probabilistic model for an event

[5] `http://en.wikipedia.org/wiki/Smoothing` (accessed 11 June, 2013)

[6] `http://nlp.stanford.edu/~wcmac/papers/20050421-smoothing-tutorial.pdf` (accessed 11 June, 2013)

e, which is a distribution $P(e)$ over an event space E. Now, we want to estimate the parameters of our model from the data. In this case, we would like to use Maximum Likelihood (ML) estimate and our model parameters will be,

$$P_{ML}(x) = \frac{c(x)}{\sum_e c(e)}$$

But the problem with the above parameter estimation is, if we have insufficient data, there will be many events x such that $c(x) = 0$. This will also make our ML estimate, $P_{ML}(x) = 0$. Most of the NLP problems have unbounded event space E. In this case, above problem is undesirable. If this problem remains while modeling, it can lead to disastrous results. Consider the case of a language model which gives zero probability to unseen words. Just because an event has never been observed in the training data, it does not mean it cannot occur in the test data. Now the question which needs to be answered is: *What should $p(x)$ be, if $c(x) = 0$?*

We need some technique which will resolve the above issue. **Smoothing** is one of the techniques, which addresses this problem. In Chen and Goodman (1996), it has been said that,

> *Whenever data sparsity is an issue, smoothing can help performance, and data sparsity is almost always an issue in statistical modeling. In the extreme case where there is so much training data that all parameters can be accurately trained without smoothing, one can almost always expand the model, such as by moving to a higher n-gram model, to achieve improved performance. With more parameters data sparsity becomes an issue again, but with proper smoothing the models are usually more accurate than the original models. Thus, no matter how much data one has, smoothing can almost always help performance.*

In the next subsection, we give an example of the sparsity problem and its solution from Chen and Goodman (1996) in the settings of n-gram language modeling.

2.3 An Example

Consider the case of a bigram model. Let us say, we have three sentences from which we want to estimate bigram language model. Consider the following examples:

john read moby dick
mary read a different book
she read a book by cher

Now the maximum likelihood model for bigram language model would be:

$$p(s) = \prod_{i=1}^{l+1} p(w_i|w_{i-1}) \tag{11}$$

Where,

$$p(w_i|w_{i-1}) = \frac{c(w_{i-1}w_i)}{\sum_{w_i} c(w_{i-1}w_i)} \tag{12}$$

Here, s represents a sentence and l is length of a sentence. Now, let us calculate the probabilities of some sentences by using above three sentences as training data. We will use Equation-11 for calculating probability values. Let us first calculate the probability of a sentence *john read a book* (# and $ are two special symbols, which are used as starting and ending symbol in the sentence respectively).

$$
\begin{aligned}
& p(\#\ john\ read\ a\ book\ \$) \\
&= p(john|\#) \cdot p(read|john) \cdot p(a|read) \cdot p(book|a) \cdot p(\$|book) \\
&= \frac{c(\#\ john)}{\sum_w c(\#\ w)} \cdot \frac{c(john\ read)}{\sum_w c(john\ w)} \cdot \frac{c(read\ a)}{\sum_w c(read\ w)} \cdot \frac{c(a\ book)}{\sum_w c(a\ w)} \cdot \frac{c(book\ \$)}{\sum_w c(book\ w)} \\
&= \frac{1}{3} \cdot \frac{1}{1} \cdot \frac{2}{3} \cdot \frac{1}{2} \cdot \frac{1}{2} \\
&\approx 0.6
\end{aligned}
$$

Let us take an other example of a sentence *cher read a book*:

$$
\begin{aligned}
& p(\#\ cher\ read\ a\ book\$) \\
&= p(cher|\#) \cdot p(read|cher) \cdot p(a|read) \cdot p(book|a) \cdot p(\$|book) \\
&= \frac{c(\#\ cher)}{\sum_w c(\#\ w)} \cdot \frac{c(cher\ read)}{\sum_w c(cher\ w)} \cdot \frac{c(read\ a)}{\sum_w c(read\ w)} \cdot \frac{c(a\ book)}{\sum_w c(a\ w)} \cdot \frac{c(book\ \$)}{\sum_w c(book\ w)} \\
&= \frac{0}{3} \cdot \frac{0}{1} \cdot \frac{2}{3} \cdot \frac{1}{2} \cdot \frac{1}{2} \\
&= 0
\end{aligned}
$$

Above calculation means that the probability of occurring a sentence *cher read a book* is zero, which is certainly not the case when we are working in the unbounded event space like natural language sentences. Since we have not seen this type of sentence in the training data, we can only say that the above sentence can occur with very **less probability**. It would be very harsh to say that the above sentence can not occur at all **(zero probability)**. After learning about this problem, in the next subsection, we describe a few smoothing techniques in brief.

2.4 Smoothing Techniques

2.4.1 Add-one Smoothing

In this technique, we change the bigram probability equation as follows:

$$
p(w_i|w_{i-1}) = \frac{1 + c(w_{i-1}w_i)}{\sum_{w_i}[1 + c(w_{i-1}w_i)]} = \frac{1 + c(w_{i-1}w_i)}{|V| + \sum_{w_i} c(w_{i-1}w_i)}
$$

Here, V is the vocabulary of the corpus. This technique is also known as **Laplace smoothing**. In general, the use of this smoothing technique is less preferred because of its poor performance. If we follow this method the probability of above two sentences will be as following. We can see that how a smoothing technique prevents assignment of zero probability to unseen data.

$$p(\#john\ read\ a\ book\$)$$

$$= \frac{1+1}{11+3} \cdot \frac{1+1}{11+1} \cdot \frac{1+2}{11+3} \cdot \frac{1+1}{11+2} \cdot \frac{1+1}{11+2}$$

$$\approx 0.0001$$

$$p(cher\ read\ a\ book)$$

$$= \frac{1+0}{11+3} \cdot \frac{1+0}{11+1} \cdot \frac{1+2}{11+3} \cdot \frac{1+1}{11+2} \cdot \frac{1+1}{11+2}$$

$$\approx 0.00003$$

2.4.2 Additive Smoothing

In this method, n-gram equation is used as:

$$p(w_i|w_{i-n+1}^{i-1}) \ = \ \frac{\delta + c(w_{i-n+1}^{i})}{\delta|V| + \sum_{w_i} c(w_{i-n+1}^{i})}$$

The main ideas behind this technique is: we assume that we have seen each n-gram δ times more than we have. So in case we miss to learn about any n-grams from the training data, we still do not get its probability value as zero. Add-one Smoothing is a specific case of this technique. Gale and Church (1994) argues that this method also performs poorly.

2.4.3 Good-Turing Estimation

In this technique, the main idea is to reallocate the probability mass of n-grams that occur $r+1$ times in the training data to the n-grams that occur r times. Particular to our problem, by using this method, we reallocate the probability mass of n-grams that were seen once to the n-grams that were never seen. For each count r, we compute an adjusted count r^*, such that

$$r^* \ = \ (r+1)\frac{n_{r+1}}{n_r}$$

Where, n_r is the number of n-grams which are seen exactly r times. Then, the probability equation becomes,

$$P_{GT}(x : c(x) = r) \ = \ \frac{r^*}{N}$$

Where, $N \ = \ \sum_{r=0}^{\infty} r \cdot n_r$. The problem with this approach is that, *What if* $n_{r+1} = 0$*?*. This is common for high values of r. Ideally we should calculate r^* as:

$$r^* \ = \ (r+1) \cdot \frac{E[n_{r+1}]}{E[n_r]}$$

But the question is how do we estimate the expectation? Thus, this method requires some elaboration to be useful. However, it provides a foundation on which other smoothing methods are build.

Apart from above smoothing techniques, there are a few other smoothing techniques as well. This techniques are as follows:

- Jelinek-Mercer smoothing (Interpolation)
- Katz smoothing (Backoff)
- Written-Bell smoothing
- Absolute discounting
- Kneser-Ney smoothing

Thus, in this section, we gave a basic idea about the smoothing techniques. In the next section, we talk about differences between the smoothing techniques and the clustering approach. We also explain that why we are using clustering in our work instead of smoothing techniques.

3 Smoothing v/s Word Clustering

Smoothing techniques are very well known for solving the data sparsity problem. But one point which smoothing techniques do not tackle is dimensionality reduction. It does not achieve the dimensionality reduction, rather opposite to that, smoothing techniques increase the size of the statistical model. These techniques just accommodate unseen data into probability distribution learned from the training data during the modeling process, albeit it is not possible to completely remove the data sparsity problem for the statistical modeling with larger event space, *e.g.,* NLP applications.

Word clustering mainly performs abstraction of the data. We believe that this process of abstraction actually addresses both the problems, the data sparsity problem and the large size of the statistical model. Because of the data abstraction, the size of the features set reduces by large extent in case of NLP problems where words are used as features, *e.g.*, language model, sentiment analysis *etc.* Additionally, the probability of encountering some unseen data which is not present in the training data becomes very less because of this data abstraction achieved by clustering process. To support the abstraction, the Occam Razor hypothesis also claims that the simpler statistical model performs better. In Balamurali et al. (2011), they also show that the data abstraction process leads to improvements in the performance of sentiment analysis. This type of abstraction can be achieved easily through abstraction processes like word clustering. Thus, for our work, we choose to work with clustering techniques as described in Section-1.

Thus, after choosing clustering as our weapon to tackle the data sparsity problem, in the next section, we talk about the related work in which researchers have used word clusters as helpful features in different NLP applications.

4 Related Work

Word cluster features have been shown to be useful in various NLP tasks. When used as an additional feature with word based language models, it has been shown to improve the system performance, *viz.*,

machine translation (Uszkoreit and Brants, 2008; Stymne, 2012), speech recognition (Martin et al., 1995; Samuelsson and Reichl, 1999), dependency parsing (Koo et al., 2008; Haffari et al., 2011; Zhang and Nivre, 2011; Tratz and Hovy, 2011), and NER (Miller et al., 2004; Faruqui and Padó, 2010; Turian et al., 2010; Täckström et al., 2012). In the subsequent subsections, we give an overview of the existing work for a few NLP applications which use word clusters.

4.1 Word Clustering for Machine Translation

In Uszkoreit and Brants (2008), they investigate the effects of applying word cluster techniques to higher-order n-gram models trained on large corpora in MT. They also introduce a modification of the exchange clustering algorithm (Kneser and Ney, 1993), which is referred as predictive exchange algorithm. The distributed version of this algorithm efficiently obtains automatic word classifications for large training corpora. They use this resulting clusterings in training partially class-based language models for MT. We have already explained the predictive exchange algorithm in detail in Section-1.5.

When using this algorithm on large corpora, even this predictive exchange algorithm would still require several days if not weeks of CPU time for a sufficient number of iterations. To overcome this, a novel distributed exchange algorithm based on the predictive exchange algorithm was introduced. The idea is basically to randomly partition the vocabulary into sets of roughly equal size and distribute the clustering work amongst multiple CPU workers. MapReduce programming model (Dean and Ghemawat, 2008) was used to implement this distributed algorithm.

The experiments were performed for English-Arabic language pair. Predictive class-based 5-gram models were trained using clusterings with 64, 128, 256, and 512 clusters on the English data. These class-based models were added as additional features to the log linear model of the Arabic-English machine translation system along with the word-based 5-gram model. It was observed that adding the class-based models leads to small improvements in BLEU score (Papineni et al., 2002). It was shown that class-based models trained using the word classifications can improve the quality of a state-of-the-art machine translation system as indicated by the BLEU score. It was also shown that class-based language models are effective tools to ease the effects of data sparsity.

Part-of-speech tagging has previously been successfully used for learning reordering rules that can be applied before training and translation. In Stymne (2012), authors investigate if clustered word classes can be used in a preordering strategy, where the source language is reordered prior to training and translation. In this work, it is shown that word clusters can be used for learning rules with only slightly worse performance than for standard POS-tags on an English–German translation task. Since the suggested preordering algorithm with word classes is fully unsupervised, the method can be applied to less-resourced languages where no taggers or parsers are available. This is not the case for many preordering methods which are based on POS-tags or parse trees. They show the usefulness of this approach for translation from resource poor language Haitian Creole to English, where the proposed approach is significantly better than the baseline.

4.2 Word Clustering for Speech Recognition

In Martin et al. (1995), authors describe an efficient method for obtaining word classes for class language models. The method employs an exchange algorithm using the criterion of perplexity improvement. Using this method, experiments were conducted on speech recognition system and improvements were observed. In Samuelsson and Reichl (1999) also, authors propose a new approach for generating a class-based language model based on part-of-speech ambiguity classes. They explored two methods for combining the class-based and word-based language models. It was also showed that both approaches led to significant reductions in word error-rate in large-vocabulary speech-recognition tasks.

4.3 Word Clustering for Dependency Parsing

For dependency parsing, in Koo et al. (2008), authors present a simple and effective semi-supervised method for training. The authors introduce features that incorporate word clusters derived from a large unannotated corpus. They show that the cluster-based features improve performance across a wide range of conditions. In the case of English unlabelled second-order parsing, performance improved from a baseline accuracy of 92.02% to 93.16%. For Czech unlabelled second-order parsing, performance improved from a baseline accuracy of 86.13% to 87.13%. It has been also shown that their method improves performance even when the small amounts of training data is available.

In Haffari et al. (2011), authors combine multiple word representations based on semantic clusters extracted from the Brown clustering algorithm and syntactic clusters obtained from the Berkeley parser to improve discriminative dependency parsing in the MST-Parser framework of McDonald et al. (2005). A method for combining diverse cluster-based models was also introduced and improvement in the accuracy of discriminative dependency parser from 90.82% to 92.13% was shown. In Täckström et al. (2012), they propose an algorithm to generate cross-lingual word clusters. They show that by augmenting the delexicalized direct transfer system of McDonald et al. (2011) with cross-lingual cluster features, they can reduce its error by up to 13% relative.

4.4 Word Clustering for Name Entity Recognition

In Miller et al. (2004), they present a technique for augmenting annotated training data with hierarchical word clusters, automatically derived from a large unannotated corpus. They encode the cluster membership in features that are incorporated in an NE tagging model. In experiments with Named Entity Recognition (NER), they show that compared to a state-of-the-art HMM-based tagger, the presented technique requires only 13% as much annotated data to achieve the same level of performance. Given a large annotated training set of 10,00,000 words, the technique achieves a 25% reduction in error over the state-of-the-art HMM trained on the same material.

Faruqui and Padó (2010) presented a study on training and evaluating a Named Entity Recognizer for German. The NER system propose by them, applies semantic generalizations (clustering) learned from a

large unlabelled corpus in the absence of large training corpora for German. Because of the generalization process, even though small corpora yield a significant improvement. In Turian et al. (2010), authors found that word clusters obtained from Brown clustering and word embeddings both can improve the accuracy of a near-state-of-the-art supervised NLP system like NER.

In Täckström et al. (2012), authors show that monolingual word cluster features induced from a large corpora helps the semi-supervised method for structure learning methods. They show that this method is robust across thirteen languages for dependency parsing and four languages for NER. A method is introduced in which cross-lingual word cluster features are used to transfer linguistic structure from English to other languages. They used a probabilistic model combining monolingual data in two languages and parallel data using which enforce the cross-lingual word-cluster constraints. They show that by applying the cross-lingual cluster features to direct-transfer NER, a relative error reduction of 26% was achieved.

Thus, in this repoer, we talked about the existing word clustering algorithms, smoothing techniques and why we choose clustering for our work instead of smoothing. We also talked about some of the similar research work where word cluster features are used in NLP applications.

References

A. R. Balamurali, Aditya Joshi, and Pushpak Bhattacharyya. Harnessing wordnet senses for supervised sentiment classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Stroudsburg, PA, USA, 2011.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, 1992.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318, Stroudsburg, PA, USA, 1996.

Pádraig Cunningham. Dimension reduction. In *Machine Learning Techniques for Multimedia*, pages 91–112. Cognitive Technologies, 2008.

Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.

Manaal Faruqui and Sebastian Padó. Training and Evaluating a German Named Entity Recognizer with Semantic Generalization. In *Proceedings of KONVENS 2010*, Saarbrücken, Germany, 2010.

Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

William A. Gale and Kenneth W. Church. What's wrong with adding one. In *Corpus-Based Research into Language. Rodolpi*, 1994.

Gholamreza Haffari, Marzieh Razavi, and Anoop Sarkar. An ensemble model that combines syntactic and semantic clustering for discriminative dependency parsing. In *Proceedings of ACL-HLT 2011*, pages 710–714, Stroudsburg, PA, USA, 2011.

Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

Reinhard Kneser and Hermann Ney. Improved clustering techniques for class-based statistical language modelling. In *Third European Conference on Speech Communication and Technology*, 1993.

Terry Koo, Xavier Carreras, and Michael Collins. Simple semi-supervised dependency parsing. In *Proceedings of ACL-HLT 2008*, pages 595–603, Columbus, Ohio, 2008.

Percy Liang. Semi-supervised learning for natural language. M. eng. thesis, Massachusetts Institute of Technology, 2005.

Sven Martin, Jörg Liermann, and Hermann Ney. Algorithms for bigram and trigram word clustering. In *Speech Communication*, pages 1253–1256, 1995.

Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98, Stroudsburg, PA, USA, 2005.

Ryan McDonald, Slav Petrov, and Keith Hall. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the Conference on EMNLP*, pages 62–72, Stroudsburg, PA, USA, 2011.

Scott Miller, Jethran Guinness, and Alex Zamanian. Name tagging with word clusters and discriminative training. In *Proceedings of HLT-NAACL 2004: Main Proceedings*, pages 337–342, Boston, Massachusetts, USA, 2004.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Stroudsburg, PA, USA, 2002.

C. Samuelsson and W. Reichl. A class-based language model for large-vocabulary speech recognition extracted from part-of-speech statistics. In *Proceedings of ICASSP 1999*, pages 537–540, 1999.

Sara Stymne. Clustered word classes for preordering in statistical machine translation. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*, pages 28–34, 2012.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 477–487, Montréal, Canada, 2012.

Stephen Tratz and Eduard Hovy. A fast, accurate, non-projective, semantically-enriched parser. In *Proceedings of EMNLP 2011*, pages 1257–1268, Stroudsburg, PA, USA, 2011.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL 2010*, pages 384–394, Stroudsburg, PA, USA, 2010.

Jakob Uszkoreit and Thorsten Brants. Distributed word clustering for large scale class-based language modeling in machine translation. In *Proceedings of ACL-08: HLT*, Columbus, Ohio, 2008.

Yue Zhang and Joakim Nivre. Transition-based dependency parsing with rich non-local features. In *Proceedings of ACL-HLT 2011*, pages 188–193, Stroudsburg, PA, USA, 2011.