Grouping Words with Semantic Diversity

Karine Chubarian[§] = Abdul Rafae Khan[†]= Anastasios Sidiropoulos[§] = Jia Xu[†] [*]

[†]Department of Computer Science, Stevens Institute of Technology
{akhan4, jxu70}@stevens.edu

[§]Department of Computer Science and Technology, University of Illinois at Chicago
{kchuba2, sidiropo}@uic.edu

Abstract

Deep Learning-based NLP systems can be sensitive to unseen tokens and hard to learn with high-dimensional inputs, which critically hinder learning generalization. We introduce an approach by grouping input words based on their semantic diversity to simplify input language representation with low ambiguity. Since the semantically diverse words reside in different contexts, we are able to substitute words with their groups and still distinguish word meanings relying on their contexts. We design several algorithms that compute diverse groupings based on random sampling, geometric distances, and entropy maximization, and we prove formal guarantees for the entropy-based algorithms. Experimental results show that our methods generalize NLP models and demonstrate enhanced accuracy on POS tagging and LM tasks and significant improvements on medium-scale machine translation tasks, up to +6.5 BLEU points. Our source code is available at https://github.com/abdulrafae/dg.

1 Introduction

Natural Language Understanding has seen remarkable success with the rise of Deep Learning. However, human languages' variety and richness result in high-dimensional inputs to NLP models, increasing learning complexity and error rates. First, open-vocabulary inputs inevitably bring rare and out-of-Vocabulary words (OOVs). Second, network complexity increases with input dimension, specifically the "curse of dimensionality" makes learning difficult on medium and small datasets.

This paper addresses these limitations by introducing new grouping methods to compute alternative language representations that simplify textual inputs. We currently have alternative language representations, such as Pinyin, Metaphone, logogram, and Emoji, that exist in natural languages and that have been shown to improve various NLP applications (Du and Way, 2017; Liu et al., 2018; Khan et al., 2020). While these representations can help, they are not developed for NLP performance. Our goal is to design algorithms for computing new language representations specifically to enhance NLP performance in this work. We ask:

"Can we compute a generalized language representation to improve NLP applications?"

An intuitive approach to answering this question is to group similar words in training and test sets and replace each word with its group. A word grouping viewed as a many-to-one mapping function can significantly reduce the vocabulary size that lowers the input feature dimensions leading to a generalized NLP model learning.

For example, let us take two sentences: (a) "you ask me."; (b) "she tells me." There are five words "ask", "she", "tells", "me", and "you" in the vocabulary. Grouping words into "A" and "B" will reduce the vocabulary size to two, resulting in a simplified language representation. We can apply conventional word clustering to group words after embedding words into a vector space and measuring their distances with cosine similarity. However, clustering can map different sentences to the same sequence of groups, making them indistinguishable. In our example, we cluster similar pronouns "you", "she" and "me" into one group indicated by "A", and verbs "ask", "tells" by "B". Then both sentences are rewritten as "A B A." The distinct meanings of the two original sentences are lost. However, if we group diverse semantic words, namely, "you", "tell" as "A"; "she", "me", "ask" as "B", then we maintain two samples of (a) "A B B." and (b) "B A B." So, the distinct meanings of the two sentences are retained. This example illustrates the need to group words so that each sentence is uniquely represented. Now, to generalize this idea,

"How can we design an algorithm that simplifies

[*]Author ordered alphabetically. Jia Xu is the corresponding author of this paper.

language representation while preserving meaning expressiveness? "

Our key observation is that the context of semantically diverse words varies more than that of semantically close words. In our approach, we measure semantic similarity using the cosine of word embeddings, learned based on context, see (Mikolov et al., 2013; Bojanowski et al., 2017; Pennington et al., 2014). Thus, similar contexts indicate semantic similarity and vice versa. In this way, our diverse grouping uses context to distinguish words from the same group, leading to a more expressive representation.

In this paper, we introduce five novel algorithms in three types that group semantically diverse words together. We develop novel theoretical methods for diverse grouping and port them in our NLP context. We begin by considering random sampling grouping. Next, we develop a grouping algorithm based on geometric distances by designing an algorithm that computes a partition of a set of points in some metric space to maximize the sum of intra-group distances. This approach is essentially the opposite of the objectives used in clustering problems, such as k-means (Forgy, 1965) and k-medians (Jain and Dubes, 1988), where one seeks to minimize a monotonic function of intra-group distances. Finally, we present a grouping algorithm to maximize diversity by maximizing the unigram entropy of the representation.

We show that the unigram entropy algorithm is $\frac{C-1}{4C+4\varepsilon C}$-approximations of the optimal solutions of maximizing the entropy of the new representation, where C is the number of groups, and ε is a small positive real number. This bound means that in the worst case, our algorithm is about 1/4 away from the optimal, while in typical cases, it could be very close to the optimal. Importantly, our theoretical results' outcomes show their usefulness in NLP tasks after we appropriately adjust them. In our experiments, each of the above methods significantly enhances the NMT accuracy by up to 6.5 BLEU points (36.9% relatively).

Our contribution can be summarized as follows:

1. *Diversity Grouping Algorithms.* We introduce various algorithms that group semantically diverse words together based on random sampling, geometric distances, and entropy maximization (§3).

2. *Formal guarantees.* We provide provable guarantees for our entropy-based algorithm (§4).

3. *Applications in NLP.* Importantly, we apply the above algorithms to NLP applications, and we show that they significantly enhance prediction accuracy. (§5).

2 Related work

While typical word clustering (Baker and McCallum, 1998; Martin et al., 1998; Feng et al., 2020) (or word class (Halteren et al., 2001)) methods collect similar words together, while our method groups *semantically diverse* words together. However, unlike the common use of clustering to smooth unseen words, our goal is to deduce the input sentence's dimension by grouping diverse words so that a word-group sequence uniquely represents a word sequence.

Our diverse grouping approach is also close to the sparse representation (Wright et al., 2008), which makes the network parameter matrices sparse without changing its dimension. Our methods reduce the Neural Network (NN) input dimension.

Such a dimension reduction can be seen as a kind of regularization on NNs. There have been many types of NN regularization methods. (Louizos et al., 2018) adds a parameter norm penalty to the objective function, (Bertsekas, 2014) adds constrained optimization, and many works exploit the sub-structure of network models, such as dropout, early stopping, and weight decay. Those approaches are very popular but may limit the capacity of models, while our methods benefit from in-domain linguistic knowledge. Work such as (Wang et al., 2018) adds augmented data (e.g., noisy data, pseudo data, etc.) but is a domain-dependent approach that inventively increases the training time.

Our work is perpendicular to the successful research in word embedding, whereby a word is mapped one-one onto a real number vector trying to preserve word pair distances. In contrast, our methods map many words into one group in a discrete space. Also, our systems build on BPE, but we do not decompose and recombine words. Therefore, our methods are additive to any improved word embedding (May et al., 2019), or BPE (Provilkov et al., 2020) versions.

Different from the inspiring work that uses Soundex, NYSIIS, Metaphone, logogram (Khan et al., 2020), Pinyin (Du and Way, 2017; Liu et al., 2018), skip-ngram (Bojanowski et al., 2017), and Huffman coding (Chitnis and DeNero, 2015; Khan

Algorithm 1 Random Grouping

Input: Vocabulary of words $\mathbb{V}$, a phonetic encoding

Parameter:

Output: Grouping γ

```
1: Perform a phonetic encoding (e.g. Metaphone) as baseline
2: Initialize the current group $i \leftarrow 1$
3: for each unique phonetic encoding do
4:     $k \leftarrow$"how many words are mapped"
5:     Sample $\{v_j\}_{j=1}^{k}$ from $\mathbb{V}$ uniformly at random
6:     Set $\gamma(v_1) \leftarrow \gamma_i, \dots, \gamma(v_k) \leftarrow \gamma_i$
7:     Set $\mathbb{V} \leftarrow \mathbb{V} \setminus \{v_1 \dots, v_k\}$
8:     $i \leftarrow i + 1$
9: end for
10: return $\gamma$
```

Algorithm 2 Poisson/Gaussian-based Random Grouping

Input: Vocabulary of words $\mathbb{V}$, a phonetic encoding

Parameter: Groups $[\gamma_1, \dots, \gamma_C], C \in \mathbb{N}$

Output: Grouping γ

```
1: for $1 \le i \le C$ do
2:     Randomly sample the group size $k$ from Poisson/Gaussian distribution (which is trained on the English Metaphone distribution)
3:     Sample $\{v_j\}_{j=1}^{k}$ from $\mathbb{V}$ uniformly at random
4:     Set $\gamma(v_1) \leftarrow \gamma_i, \dots, \gamma(v_k) \leftarrow \gamma_i$
5:     Set $\mathbb{V} \leftarrow \mathbb{V} \setminus \{v_1 \dots, v_k\}$
6: end for
7: return $\gamma$
```

et al., 2020), our study aims to develop new artificial algorithms that lower the dimensions of the textual inputs with smaller vocabularies.

3 Algorithms for Diverse Grouping

We now present our algorithms. We denote the set of words by $\mathbb{V}$, and the set of groups by $\mathcal{V}$, i.e., $\mathcal{V}$ is a subset of the powerset of $\mathbb{V}$. Each grouping can be encoded as a function γ that maps each word $w \in \mathbb{V}$ to some group $\gamma(w) \in \mathcal{V}$.

3.1 Random Grouping

Our first approach computes a random grouping, as shown in Algorithm 1. This algorithm's complexity is $\mathcal{O}(\mathbb{V})$, where V is vocabulary size.

We map each word to a group chosen uniformly at random. C is a hyperparameter indicating the total number of groups. Because it is expensive to tune C with exhaustive search, we set C as the total number of Metaphones in English, inspired by previous work (Du and Way, 2017; Liu et al., 2018) in which phonetics improves NMT in specific languages. Furthermore, each group's size follows the natural phonetic encoding distribution (e.g., Metaphone (Philips, 1990)) by considering each phonetic encoding as a group. For example, each Metaphone is considered as a group, and the number of groups in the random grouping is set to the number of unique Metaphone.

Algorithm 2 extends Algorithm 1 by learning a Poisson or Gaussian model for the distribution of group sizes. We fit the distribution of the Metaphone group sizes into a Poisson or Gaussian distribution. Then, we sample the group size according to this Poisson or Gaussian distribution. Finally, we sample words for each group uniform randomly.

3.2 Distance-Based Diverse Grouping

We now introduce our grouping algorithm, which uses distances on the vector representations of words. The complexity of this algorithm is $\mathcal{O}(\mathbb{V}^2)$.

Our approach is described in Algorithm 3, which is inspired by the classical 2-approximation algorithm for k-center clustering (Gonzalez, 1985). Our algorithm works as follows: randomly pick a word from the vocabulary and add it to the list L'. Pick the second word that is the furthest from the first word, pick the third word which is furthest from the closest of the two selected words, and so on. Finally, for each group size k that follows a Metaphone encoding size distribution, group the top k words into group one and remove those k words from the list. This process is performed iteratively until all words are assigned. We use cosine-similarity to measure the pairwise distance between words. Figure (1) illustrates the work of the algorithm.

3.3 Maximum Entropy-Based Unigram Diverse Grouping

We now present our grouping algorithm, which maximizes unigram entropy. Our ultimate goal is to maximize the information kept (or reduce the infor-

Algorithm 3 Distance-Based Diverse Grouping

Input: Vocabulary of words $\mathbb{V}$
Parameter: Groups $[\gamma_1, \dots, \gamma_C]$ with sizes k_i
Output: Grouping γ

```
 1: Embed $\mathbb{V}$ in $\mathbb{R}^N$ using e.g. word2vec
 2: $\mathbb{W} \leftarrow$ resulting embedding
 3: Randomly pick $w_0 \in \mathbb{W}$, append $w_0$ to the ranked list $L'$
 4: for $1 \le j \le |\mathbb{V}|$ do
 5:     $maxmin = 0$
 6:     for all $w_i \in \mathbb{W} \setminus L'$ do
 7:         Find $mindist_i \leftarrow \min_{v \in L'} \|w_i - v\|_2$
 8:         if $mindist_i > maxmin$ then
 9:             Set $maxmin \leftarrow mindist_i$, $W \leftarrow w_i$
10:         end if
11:     end for
12:     Append $W$ in $L'$
13: end for
14: Perform a phonetic encoding (e.g. Metaphone)
15: $i \leftarrow 0$
16: for each encoding & $i++$ do
17:     Assign the encoding size as $k_i$
18: end for
19: for $1 \le i \le C$ do
20:     Set $\gamma(v) \leftarrow \gamma_i$ for the top $k_i$ points of $L'$
21:     Remove the top $k_i$ points from $L'$
22: end for
23: return $\gamma$
```

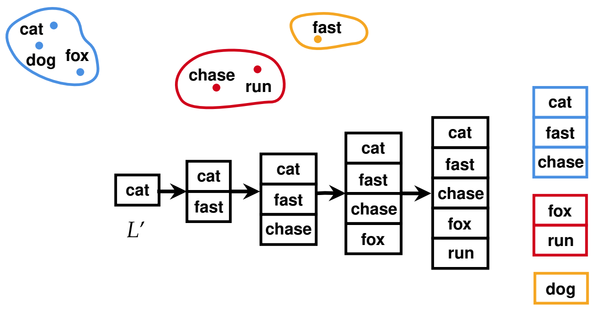

Figure 1: An example of Distance-Based Diverse Grouping

mation loss) from the original input sentences in the newly coded sentences with a reduced vocabulary. Distance-based diverse grouping does not consider the probability (relative frequency) of each element (original word), i.e., the input distribution. For example, if a word occurs very frequently, e.g., "the", which can be followed by many words (different nouns), then the context of "the" cannot help much to distinguish its meaning. Therefore, it is less ambiguous to assign a frequent word ("the") than an infrequent word to a unique codeword without sharing the codeword. Shannon entropy provides the quantitative measure on information considering such an input distribution.

Importantly, Maximum Entropy-Based Unigram Diverse Grouping (Entropy) is a more efficient algorithm, with a complexity of $N^{\mathcal{O}(1)}$, where N is the number of running words in training. Furthermore, we provide a provable guarantee of about $\frac{1}{c}$-approximation.

The entropy-based diverse grouping aims to maximize the diversity of group assignments in the given text with respect to its entropy. Because the entropy is maximal when the underlying distribution is as close to uniform as possible, this objective captures the diversity requirement. As an illustration, consider the following text: (1) "she is running very fast."; (2) "he is running very fast."; (3) "running is very popular today." We want to form three groups. This text has a length fourteen; thus, a grouping with high entropy aims to keep the frequency of each group around $\frac{5}{14}$. Therefore, the frequent words like "running" and "is" are likely to be grouped apart; infrequent words will be spread among groups uniformly. For instance, the grouping $\{1\text{: running, fast, late}\}$, $\{2\text{: is she today}\}$, $\{3\text{: he very popular}\}$ has a group frequency of $\frac{5}{14}$, $\frac{5}{14}$, $\frac{4}{14}$ hence achieving high entropy. Furthermore, the grouped words appear to be diverse enough so that each pair of groups $\{11, 12, \dots, 33\}$ appears exactly once or twice after we perform the grouping. We consider the entropy with respect to a distribution induced by the *relative frequencies* of *group unigrams*.

Formally, for any group γ_i we can define *a relative frequency of a group* as

$$c_{\gamma_i} = \sum_{w \in \mathbb{V}:\, \gamma(w) = \gamma_i} F_w$$

where F_w is a relative frequency of a word w. If the group is empty, its relative frequency is 0. The *unigram entropy* of a grouping with $\mathcal{V} = [\gamma_1, \dots, \gamma_C]$ is

$$H(\gamma) = -\sum_{i=1}^{C} c_{\gamma_i} \log c_{\gamma_i} \qquad (1)$$

We are interested in a grouping that maximizes (1).

3.3.1 Algorithms for Unigram Entropy Diverse Grouping

We show how to compute a grouping for (1) by adapting the approximation algorithm for submodular maximization under matroid constrains due to (Lee et al., 2009). In our terminology, their algorithm applies three operations to all possible pairs (w, γ_i) where $w \in \mathbb{V}$ and $\gamma_i \in \mathcal{V}$. It terminates when for every (w, γ_i) each operation is either impossible to perform or the resulting entropy gain is below $(1 + \varepsilon/(C|\mathbb{V}|)^4)H_{\text{old}}$ where H_{old} denotes an entropy of the grouping before the operation. These operations are:

1. Put a word w into a group γ_i
2. Remove a word w from a group γ_i
3. Remove a word w from a group γ_i and then put another word v into a group γ_j (we allow either $w = v$ or $\gamma_i = \gamma_j$).

After we find the initial grouping, some of the words may remain unassigned. We note that in general, adding new words to a grouping may decrease the entropy. As an example, assume that we have two groups γ_1, γ_2 with $c_{\gamma_1} = 0.25, c_{\gamma_2} = 0.5$ and an ungrouped word w with $F_w = 0.25$. The current entropy of a grouping is $-0.25 \log 0.25 - 0.5 \log 0.5$. Setting $\gamma(w) = \gamma_2$ means that the contribution of c_{γ_2} is now $-0.75 \log 0.75 < -0.5 \log 0.5$ hence the total entropy decreased. To minimize the potential entropy loss, we map ungrouped words to a group γ_j with the smallest partial entropy $G(c_{\gamma_j}) = -c_{\gamma_j} \log(c_{\gamma_j})$.

Algorithm 5 explains the detail of the unigram entropy diverse grouping algorithm, respectively. We give proofs to this algorithm with the main result stated as follows:

Theorem 1. *Given any precision parameter $\varepsilon > 0$, Algorithm 5 runs in polynomial time, and computes a grouping that is a $\frac{C-1}{4C+4\varepsilon C}$-approximation to the maximum unigram entropy.*

Roughly speaking, our algorithms are about 1/4 away from optimal of maximizing the entropy. In typical cases, our algorithms could be very close to the optimal. Section 4 describes the details of the proofs.

4 Entropy Maximization as Submodular Maximization

In order to apply the optimization techniques from (Lee et al., 2009), which we briefly describe in Section 3.3.1, we need to use a different representation of grouping. We view a grouping γ as the set of all pairs (w, γ_i) where $\gamma(w) = \gamma_i$. For

Algorithm 4 Initial grouping (Lee et al., 2009)

```
Input: Vocabulary of words $\mathbb{V}$, relative frequencies $F_w$
Parameter: Groups $[\gamma_1, \dots, \gamma_C], C \in \mathbb{N}$, precision parameter $\varepsilon$
Output: Grouping $\gamma' : \mathbb{V} \to \mathcal{V}$

 1: Brute-force search for $w_0$ with the biggest partial entropy
        $w_0 \in \arg\max_{w \in \mathbb{V}} \{-F_w \log F_w\}$
 2: Assign $\gamma'(w_0) \leftarrow \gamma_1$
 3: Set threshold $t \leftarrow 1 + \varepsilon/(C|\mathbb{V}|)^4$
 4: until no update is possible do:
 5: Try all possible updates and all pairs $(w, \gamma_i)$:
 6: Update 1 Add $w$ to $\gamma_i$, compute entropy of the update $H_1$
 7: Update 2 Remove $w$ from $\gamma_i$, compute entropy of the update $H_2$
 8: Update 3 Remove arbitrary $v$ from $\gamma(v)$, add $w$ to $\gamma_i$, compute entropy of the update $H_3$
 9: if Update j can be used on $(w, \gamma_i)$ then
10:     if Updated entropy $H_j > tH(\gamma)$ then
11:         Perform update j
12:     end if
13: end if
14: return $\gamma$
```

a vocabulary $\mathbb{V}$ and groups $\mathcal{V}$ we denote a set of all possible pairs (w, γ_i) as $\mathbb{V} \times \mathcal{V}$. For instance, let $\mathbb{V} = \{$“she”, “tells”, “me”$\}$ and $\mathcal{V} = [\gamma_1, \gamma_2]$. Then $\mathbb{V} \times \mathcal{V} = \{$(“she”, 1), (“she”, 2), (“tells”, 1), (“tells”, 2), (“me”, 1), (“me”, 2)$\}$. Consider a grouping γ such that γ(“she”) $= \gamma$(“tells”) $= 1$ and γ(“me”) $= 2$. Then γ can be described as a set is $\{$(“she”, 1), (“tells”, 1), (“me”, 2)$\}$. We say that such set *defines a grouping* and refer to a family of all such sets as *grouping set family*.

Note that an arbitrary set in $\mathbb{V} \times \mathcal{V}$ may not define a grouping. For instance, the set $\{$(“she”, 1), (“tells”, 1), (“tells”, 2), (“me”, 2)$\}$ does not, as it maps “tells” to more than one group.

To apply the results of (Lee et al., 2009), we need to show that the grouping set family forms a *matroid* (Lee et al., 2009) on $\mathbb{V} \times \mathcal{V}$.

Lemma 1. *The grouping set family defines a matroid on $\mathbb{V} \times \mathcal{V}$.*

To satisfy the conditions of a matroid, we need

Algorithm 5 Unigram Entropy Diverse Grouping

Input: Vocabulary of words $\mathbb{V}$, relative frequencies F_w

Parameter: Groups $[\gamma_1, \ldots, \gamma_C], C \in \mathbb{N}$, precision parameter ε

Output: Grouping $\gamma : \mathbb{V} \to \mathcal{V}$

```
1: Compute initial grouping $\gamma'$ using Algorithm (4)
2: if $\gamma'(w)$ is undefined for some $w$ then
3:     Let $\mathbb{W} \leftarrow \{w \in \mathbb{V} : \gamma'(w)$ is undefined$\}$
4:     Create a new grouping $\gamma \leftarrow \gamma'$
5:     Find a group $\gamma_{i_0}$ with the lowest partial unigram entropy:
           $$\gamma_{i_0} \in \arg\min_{\gamma'_j \in \mathcal{V}} \left\{ G\left(c_{\gamma'_j}\right) \right\}$$
6:     for all $w \in \mathbb{W}$ do
7:         Set $\gamma(w) \leftarrow \gamma_{i_0}$
8:     end for
9: end if
10: return $\gamma$
```

two properties. As an example[1], consider $\mathbb{V}$ and $\mathcal{V}$ as in Figure (2). Firstly, let $Q \subseteq \mathbb{V} \times \mathcal{V}$ be a set that defines a grouping of $\mathbb{V}$. For instance, $Q = \{$(point,1), (graph, 1), (noun, 1), (text, 2), (science, 2)$\}$. Then every $R \subset Q$ such as $R = \{$(point,1), (graph, 1), (text, 2)$\}$ must define a grouping as well. Secondly, take two sets $S, T \subseteq \mathbb{V} \times \mathcal{V}$ that both define groupings. Then if $|T| < |S|$, we should always be able to find a pair $(w, \gamma_i) \in S \setminus T$ such that adding (w, γ_i) to T results in a grouping. In Figure (2), this pair is (point, 1) in S; $T \cup$ (point, 1) does define a new groping.

Moreover, the algorithm from (Lee et al., 2009) requires an objective function to be *submodular* (Lee et al., 2009). Intuitively, submodularity means that a function value changes less for larger inputs.

Lemma 2. *The function $H : \mathbb{V} \times \mathcal{V} \to \mathbb{R}$ is non-negative and submodular.*

To see that H is submodular, consider $\mathbb{V}$ and $\mathcal{V}$ as pictured in Figure (2). Consider groupings γ and γ' induced by R and Q from Figure (2). Assume that we add $(\text{word}, 1)$ to γ and γ'. Relative frequencies of γ_2 remains unchanged for γ, γ'. Then the entropy gain for γ' and γ depends only on the

[1]The full proof is provided in Appendix.

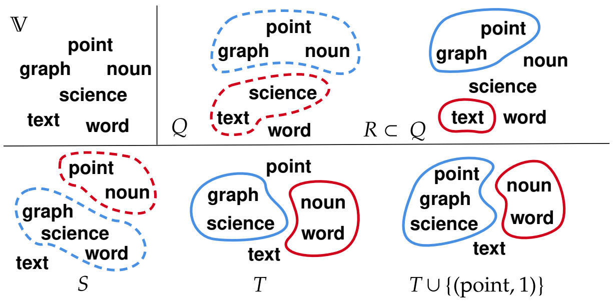

Figure 2: Properties 2) and 3) of a matroid for a grouping with $\mathbb{V} = \{$graph, point, noun, word, text, science$\}$ and $\mathcal{V} = [1, 2]$. Blue color denotes group 1, red color denotes group 2.

partial entropies of a group indexed by 1:

$$-\left(c_{\gamma'_1} + F_{\text{word}}\right) \log\left(c_{\gamma'_1} + F_{\text{word}}\right) + c_{\gamma'_1} \log(c_{\gamma'_1})$$
$$-\left(c_{\gamma_1} + F_{\text{word}}\right) \log\left(c_{\gamma_1} + F_{\text{word}}\right) + c_{\gamma_1} \log(c_{\gamma_1})$$

Every pair (w, γ_i) grouped by γ is also grouped by γ' thus $c_{\gamma_1} < c_{\gamma'_1}$. Because the function $L(x) = -x \log(x + F_{\text{well}}) + x \log x$ is monotone decreasing for all real non-negative values x, we have $L(c_{\gamma_1}) > L(c_{\gamma'_1})$. Hence, the larger grouping gains less in entropy than the smaller one.

Now we give a sketch of the proof of Theorem 1.

Proof. We claim that $H(\gamma) \geq \frac{C-1}{4C+4\varepsilon C} H^*$ where H^* is the largest unigram entropy among all groupings. We should consider the case $H(\gamma) < H(\gamma')$. The groupings γ and γ' differ only in index i_0. Thus the difference $H(\gamma) - H(\gamma')$ is equal to the difference in the partial entropies

$$G\left(c_{\gamma_{i_0}}\right) - G\left(c_{\gamma'_{i_0}}\right).$$

We note that the group γ'_{i_0} with the smallest partial entropy contributes at most $H(\gamma')/C$ to the total entropy of γ'. Moreover, partial entropy of γ_{i_0} is always non-negative. We obtain $H(\gamma) - H(\gamma') \geq -H(\gamma')/C$. Our bound follows by plugging in the estimation $H(\gamma') \geq H^*/(4 + 4\varepsilon)$ which is the approximation guarantee for the Algorithm 4 from (Lee et al., 2009). $\square$

5 Experiments

Combination Methods Below, we will discuss how to incorporate our new representation using any of our grouping methods in NLP tasks. Firstly, we group each word independently. Applying a grouping function $\gamma(\cdot)$ in Section 3 on each word $x_1, x_2, x_3, \cdots, x_i, \cdots, x_{I'}$ in an input sentence

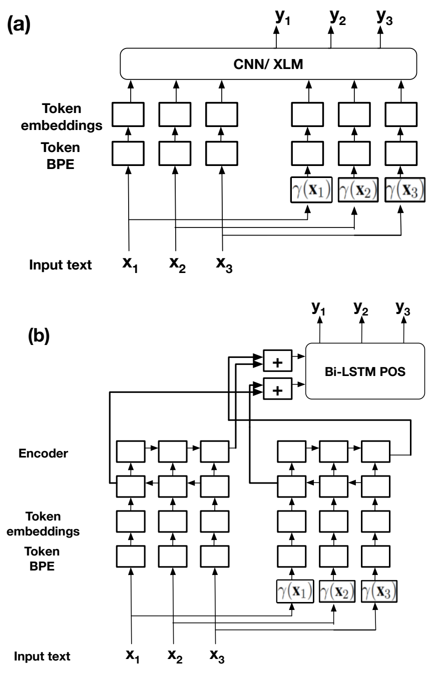

Figure 3: Combination methods for different NN architectures: (a) concatenation for ConvS2S and XLM; (b) linear interpolation on encoder outputs for Bi-LSTM with attention. $\gamma(\cdot)$ is a word grouping function.

one by one generates a sequence of word groups $\gamma(x_1), \gamma(x_2), \gamma(x_3), \cdots, \gamma(x_i), \cdots, \gamma(x_{I'})$ in the same length I'. Note that we use the term "word" loosely here; it can mean a word or a subword (of a BPE token), or even a character.

The first combination method is **concatenation**, see Figure 3a. We apply this method in NMT. First, we concatenate two input sources. Next, we apply the Byte-Pair-Encoding (Sennrich et al., 2015) (BPE) and word embeddings implemented by Řehůřek and Sojka (2010) on each word $\epsilon(x)$ and its codeword $\epsilon_\gamma(\gamma(x))$. We separately train word embedding on groups and on words. Thus, $\epsilon_\gamma(\cdot)$ and $\epsilon(\cdot)$ are different functions. As shown in Figure 3, the input to the NLP system is the embedded words of a sentence, $\tilde{\epsilon}(x_1), \tilde{\epsilon}(x_2), \tilde{\epsilon}(x_3), \cdots, \tilde{\epsilon}(x_i), \cdots, \tilde{\epsilon}(x_I)$, where $\tilde{\epsilon}(x_i)$ is the concatenation of the embedded words $\epsilon(x_i)$ and their groups $\epsilon_\gamma(\gamma(x_i))$:

$$\tilde{\epsilon}(x_i) = [\epsilon(x_i); \epsilon_\gamma(\gamma(x_i))] \tag{2}$$

The second method is **linear combination on encoder outputs**, see Figure 3b. We use this method in part-of-speech (POS) tagging. The input to the linear combiner is the grouped sentence, represented by a sequence of hidden states $\tilde{h^1}(\epsilon(x_I)), \cdots, \tilde{h^j}(\epsilon(x_I)), \cdots, \tilde{h^J}(\epsilon(x_I))$ of the last position I in each of the encoder layers $j \in [1, 2, \cdots, J]$. J is the number of nodes at each decoder layer. Recall that each hidden state is a real vector $\mathbb{R}^d$, which is why we can use the vector space operations such as addition on it. For convenience, we denote the last hidden state of the j-th encoder layer, which we take as the input to the decoder, $\tilde{h^j}(\epsilon(x_I))$, by $\tilde{h}^j_I$, the last hidden state of the j-th encoder layer of the original textual sentence $h^j(\epsilon(x_I))$ by h^j_I, and the last hidden state of the j-th encoder layer of the grouped sentence $h^j(\epsilon_\gamma(\gamma(x_I)))$ by $h_{\gamma}{}^j_I$. The combined encoder hidden state $\tilde{h^j}$ is a linear interpolation of the hidden states of the textural input and its group input:

$$\tilde{h^j} = (1-\alpha)h^j_I + \alpha h_{\gamma}{}^j_I \tag{3}$$

As shown in Figure (3b), each layer's combined last hidden state is fed into the baseline decoder with the operator of $+$. α is the encoder weight of the grouped sentence, and here, $\alpha = 0.5$.

In the following context, we will show the our methods' evaluation results on three representative NLP tasks: (1) Machine translation as a recognition and generation problem; (2) Language modeling as a regression problem; and (3) POS tagging as a typical sequence labeling problem. We will show that our methods have the potential to improve any NLP application with textual inputs.

5.1 Neural Machine Translation

Dataset We empirically verify our method on the IWSLT'17 dataset containing *226 thousand* sentences. Table 1 shows the vocabulary statistics before and after the pre-processing on the original and the concatenated data. We carry out experiments on the English-to-French (EN-FR) language direction.

We also carry out experiments on additional medium and small NMT tasks. For medium-sized tasks, we use the IWSLT'17 dataset with language directions including English to German (EN-DE), German to English (DE-EN), and English to Chinese (EN-ZH). We use the MTNT'18 dataset with language directions English to French (MTNT EN-FR) and French to English (MTNT FR-EN) for the small-sized task.

Baseline and Setup As a filter in pre-processing, every sentence is restricted to 250 characters and 1.5 length ratio between source and target sentences using Moses tokenizer (Koehn et al., 2007).

Running Words	Method	EN	FR
Before BPE	Baseline	4.8	5.1
	+Entropy	9.5	5.1
After BPE	Baseline	5.2	5.6
	+Entropy	10.2	5.8
Vocabulary	**Method**	**EN**	**FR**
Before BPE	Baseline	63	81
	+Entropy	67	81
After BPE	Baseline	11	13
	+Entropy	12	11

Table 1: IWSLT'17 English-French statistics of running words in millions (M) and vocabulary words in thousands (K).

The Byte-pair encoding model (with $16K$ BPE operations) is jointly trained on the source textual word inputs, cluster ID inputs, and target outputs. The baseline NMT model is the Convolutional Sequence to Sequence (Gehring et al., 2017) (ConvS2S), with the following parameter setting: the embedding dimension as 512, the learning rate as 0.25, the gradient clipping as 0.1, the dropout ratio as 0.2, and the optimizer as NAG. The training is terminated when the validation loss does not decrease for five consecutive epochs. For Chinese translations, we use the IWSLT post-processing script (IWSLT, 2021). Finally, the translation accuracy is measured with the BLEU score using SacreBLEU (Post, 2018).

Distance Measure To empirically compare across random and distance-based grouping algorithms, we measure the intra-group average distance of group pairs as follows: For each group in the source side vocabulary of the training and test set, compute the sum of the cosine distance $1 - \frac{A \cdot B}{||A|| \cdot ||B||}$ of the embedding of each word pair, then divide it by the total number of word pairs in this group to get the group diversity. Then, average the distance of all groups in the vocabulary. Each algorithm generates the same number (63992) of word groups. The average distance of Poisson/Gaussian-based Random Grouping is 0.1286, and that of Rank-based Diverse Grouping is 0.1291. This finding is consistent with the translation BLEU score in Table 2. The greater the intra-group distance, the higher the accuracy. Maximum entropy approaches cannot be compared with this measure because it takes the entropy as an objective function. We have provided its provable guarantee in Section 4.

Method	Test
Baseline	17.6
Word Grouping	
Poisson R.G. (Alg.2)	22.4
Gaussian R.G. (Alg.2)	23.4
R.G. (Alg.1)	23.3
Distance D.G. (Alg.3)	23.6
Entropy (Alg.4)	**24.1** (+36.9%)

Table 2: Translation results in BLEU[%] on IWSLT'17. EN-FR. Baseline is (Gehring et al., 2017) on words. Dev: IWSLT test 2013, test 2014, test 2015; Test: IWSLT test 2017. Our methods include Random Grouping (R.G.), Poisson/Gaussian-based Random Grouping (Poisson/Gaussian R.G.), Distance-based Diverse Grouping (Distance D.G.), and Entropy-based Diverse Grouping (Entropy), respectively. BPE: $16k$.

Dataset	IWSLT			MTNT
Method	EN-DE	DE-EN	EN-ZH	EN-FR
Baseline	19.4	22.6	18.6	19.4
Entropy	21.0	24.0	19.2	**23.9** (+18.8%)

Table 3: Left: Translation results in BLEU[%] on IWSLT'17 (EN-DE, DE-EN, EN-ZH; Dev: test 2013, test 2014, test 2015; Test: test 2017). Right: Translation results in BLEU[%] on MTNT'18 (EN-FR; Dev: MTNT dev; Test: MTNT test 2018) datasets. Baseline: (Gehring et al., 2017) on words. BPE operations: $16k$.

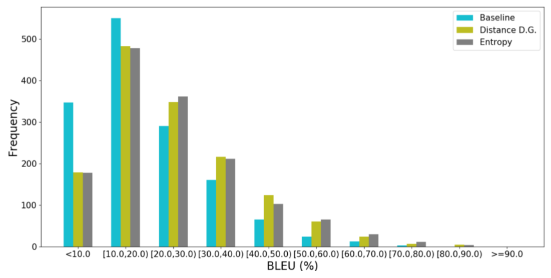

Figure 4: Histogram of Sentence-level BLEU Scores for baseline, Distance-based D.G. & Entropy.

Results For IWSLT'17 task, Table 2 shows the improvement when applying each of our methods on the ConvS2S baseline. All of our methods significantly enhance the accuracy of the NMT systems. Among them, the entropy-based diverse grouping achieves the greatest improvement, i.e., +6.5 BLEU points, which is +36.9% relative improvement.

Analysis Figure 4 compares entropy, distance-based D.G. and the baseline method with respect to the sentence-level BLEU score in a histogram (Neubig et al., 2019). The baseline method generates

Source	Qu'est-ce que cela signifie pour le procrastinateur ?
Reference	Now, what does this mean for the procrastinator?
Baseline	Well, there's an issue of what it means for procrastinator?
Entropy	Now, what does this mean for procrastinator?
Source	Pendant que ça pousse, ils font des changements.
Reference	While it's growing, they make changes.
Baseline	It's going to push it, they make change.
Entropy	So, i think it's growing, they make change.
Source	Mes parents ne se sont jamais plaints.
Reference	And you know, my parents never complained.
Baseline	It's never complained.
Entropy	So, my parents never complained.

Table 4: Comparison of translation outputs for baseline, Distance D.G. & Entropy.

almost double low-quality translations (347) compared to the distance D.G. and entropy methods (178 and 179), while the latter two methods generate many more high-quality translations with BLEU above 20%.

Table 4 shows the FR-EN baseline and entropy translation outputs, respectively. We observe that our entropy method is particularly better than the baseline when the baseline fails in: (1) performing a reasonable translation; (2) missing phrases; (3) mis-translating phrases.

5.2 POS Tagging

We evaluate our approach in POS Tagging on Brown Corpus (Francis and Kucera, 1979). Brown corpus is a well-known English dataset for POS and contains 57 341 samples. We uniform randomly sample 64% data as the training set, 16% as the validation set, and 20% as the test set. Our baseline is a Keras (Chollet, 2015) implementation (Joshi, 2018) of Bi-LSTM POS Tagger (Wang et al., 2015). We train word embedding (Mikolov et al., 2013) implemented by Řehůřek and Sojka (2010) with 100 dimensions. Each of the forward and the backward LSTM has 64 dimensions. We use a categorical cross-entropy loss and RMSProp optimizer. We also use early stopping based on validation loss.

	Dev	Test		
	Loss		Accuracy	Error Rate
Baseline	5.16	5.39	98.61	1.39
R.G.	5.32	5.07	98.69	1.31
Poisson R.G.	5.56	5.27	98.62	1.38
Entropy	5.48	5.22	98.66	1.34 (-3.60%)

Table 5: POS accuracy and error rate in [%] on Brown corpus. Baseline is Bi-LSTM POS Tagger (Joshi, 2018) on words.

	Test PPL
Baseline	22.85
Entropy	21.99 (-3.76%)

Table 6: LM PPL on the English part of IWSLT'17 EN-FR. Baseline: XLM (Lample and Conneau, 2019).

5.3 Language Modeling (LM)

We train and evaluate the English part of EN-FR IWSLT'17 dataset. We use 256 embedding dimensions, six layers, and eight heads for efficiency. We set dropouts to 0.1, the learning rate to 0.0001, and BPE operations to $32k$. We used Adam optimizer with betas of 0.9. As shown in Table 6, Entropy-based diverse grouping reduces PPL of the baseline system, i.e., 3.76% relatively.

6 Conclusion

We introduce a novel approach that generalizes Deep Learning models by grouping input words to maximize their semantic diversity. To this end, we design a family of algorithms based on random sampling, geometric distance, and entropy, and provide provable guarantees to the entropy-based diverse grouping. Our methods reduce the number of low-quality translation outputs ($< 10\%$ in BLEU) to half and greatly increase the high-quality translation ($> 20\%$ in BLEU) ratio. Experiments show that our approach significant improves over state-of-the-art baselines in Neural Machine Translation (i.e., up to +6.5 BLEU points) and achieves higher accuracy in POS Tagging and Language Modeling.

Acknowledgement

We appreciate the National Science Foundation (NSF) Award No. 1747728 and the National Science Foundation of China (NSFC) Award No. 61672524 to fund this research. We are also thankful for the support of the Google Cloud Research Program.

References

L Douglas Baker and Andrew Kachites McCallum. 1998. Distributional clustering of words for text classification. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Dimitri P Bertsekas. 2014. *Constrained optimization and Lagrange multiplier methods*. Academic press.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*.

Rohan Chitnis and John DeNero. 2015. Variable-length word encodings for neural translation models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

François Chollet. 2015. Keras. https://github.com/keras-team/keras.git.

Jinhua Du and Andy Way. 2017. Pinyin as subword unit for Chinese-sourced neural machine translation. In *Proceedings of Irish Conference on Artificial Intelligence and Cognitive Science*.

Yukun Feng, Chenlong Hu, Hidetaka Kamigaito, Hiroya Takamura, and Manabu Okumura. 2020. A simple and effective usage of word clusters for CBOW model. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*.

Edward W Forgy. 1965. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–769.

W. N. Francis and H. Kucera. 1979. Brown corpus manual. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of International Conference on Machine Learning*.

Teofilo F Gonzalez. 1985. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306.

Hans van Halteren, Jakub Zavrel, and Walter Daelemans. 2001. Improving accuracy in word class tagging through the combination of machine learning systems. *Computational Linguistics*, 27(2):199–229.

IWSLT. 2021. IWSLT post-processing toolkit. In *http://hltshare.fbk.eu/IWSLT2015/chineseText2Chars.pl*.

Anil K Jain and Richard C Dubes. 1988. *Algorithms for clustering data*. Prentice-Hall, Inc.

Aneesh Joshi. 2018. LSTM POS tagger. https://github.com/aneesh-joshi/LSTM_POS_Tagger.git.

Abdul Rafae Khan, Jia Xu, and Weiwei Sun. 2020. Coding textual inputs boosts the accuracy of neural networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of Association for Computational Linguistics*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems*.

Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. 2009. Non-monotonic submodular maximization under matroid and knapsack constraints. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 323–332. ACM.

Hairong Liu, Mingbo Ma, Liang Huang, Hao Xiong, and Zhongjun He. 2018. Robust neural machine translation with joint textual and phonetic embedding. In *Proceedings of Association for Computational Linguistics*.

Christos Louizos, Max Welling, and Diederik P Kingma. 2018. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*.

Sven Martin, Jörg Liermann, and Hermann Ney. 1998. Algorithms for bigram and trigram word clustering. *Speech Communication*, 24(1):19–37.

Avner May, Jian Zhang, Tri Dao, and Christopher Ré. 2019. On the downstream performance of compressed word embeddings. *Advances in Neural Information Processing Systems*, 32:11782–11793.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*.

Graham Neubig, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, and Xinyi Wang. 2019. compare-mt:

A tool for holistic comparison of language generation systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Lawrence Philips. 1990. Hanging on the metaphone. *Computer Language*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the Conference on Language Resources and Evaluation 2010 Workshop on New Challenges for NLP Frameworks*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *arXiv preprint arXiv:1510.06168*.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018. SwitchOut: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma. 2008. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227.

Appendix A Used notation

We list the notation used throughout the paper
$\mathbb{V}$: vocabulary of words
$\mathcal{V}$: vocabulary of groups
w, v: a word
F_w: relative frequency of a word w
γ_i, γ_j: a group
$\mathbb{V} \times \Gamma$: set of all possible pairs (w, γ_i)
c_{γ_i}: relative frequency of a group γ_i
γ: an assignment (grouping)
$H(\gamma)$: unigram entropy of a grouping γ
$G\left(c_{\gamma'_j}\right)$: partial enropy of a group γ_i
C: number of groups
$[1, \ldots, C]$ - natural numbers from 1 to C
$\mathbb{N}$ - natural numbers

Appendix B Omitted proofs

Definition 1 (Matroid)**.** *Let* Ω *be a finite set* (universe) *and* $\mathcal{I} \subseteq 2^{\Omega}$ *be a set family* (independent sets). *A pair* $\mathcal{M} = (\Omega, \mathcal{I})$ *is called* a matroid *if*

1. $\emptyset \in \mathcal{I}$
2. *If* $Q \in \mathcal{I}$ *and* $R \subseteq Q$ *then* $R \in \mathcal{I}$
3. *For any* $Q, R \in I$ *with* $|R| < |Q|$ *there exists* $\{x\} \in Q \setminus R$ *such that* $R \cup \{x\} \in \mathcal{I}$.

Let us denote a family of all grouping sets of $\mathbb{V} \times \mathcal{V}$ as $\mathcal{G}$.

Proof of Lemma 1. We have to show that $(\mathbb{V} \times \mathcal{V}, \mathcal{G})$ satisfies three condition from the Definition 1.

1. An empty grouping is a grouping.
2. Consider an arbitrary $Q \in \mathcal{G}$ and $R \subset Q$. Since Q defines a grouping, for any $(w, \gamma_i) \in Q$ we have $(w\gamma_j) \notin Q$ for all $\gamma_j \neq \gamma_i$. Therefore, for all $(w, \gamma_i) \in R$ we have $(w\gamma_j) \notin R$ given $\gamma_j \neq \gamma_i$ and thus R defines a grouping as well.
3. Consider two arbitrary $R, Q \in \mathcal{G}$ with $|R| < |Q|$. Let us denote $\{w \in \mathbb{V} : (w, \gamma_i) \in Q \text{ for some } \gamma_i\}$ as $\pi(Q)$. We claim that $|Q| = |\pi(Q)|$. Otherwise, there must exist w such that $(w, \gamma_i), (w, \gamma_j) \in Q$ and $\gamma_i \neq \gamma_j$. However, this is forbidden for a set which defines a grouping. Analogously, $|R| = |\pi(R)|$. Since both R, Q are finite, we have $0 < |Q \setminus R| = |\pi(Q)| - |\pi(R)| = |\pi(Q) \setminus \pi(R)|$. Consider

an arbitrary $w' \in \pi(Q) \setminus \pi(R)$ and its group $\gamma_{i'}$ in Q; we have $(w', \gamma_{i'}) \in Q \setminus R$. Moreover, since w' is ungrouped by R, we conclude that $R \cup \{(w', \gamma_{i'})\} \in \mathcal{G}$ and finish the proof.

□

Definition 2 (Submodular function). *A function $f : 2^{\Omega} \to \mathbb{R}$, where Ω is finite, is* submodular *if for any $X \subseteq Y \subseteq \Omega$ and any $x \in \Omega \setminus Y$ we have*

$$f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y).$$

For any non-negative real x and fixed $a > 0$, we denote $-(x+a)\log_2(x+a) + x \log x$ as $L_a(x)$.

Proof of Lemma 2. First, we show that $H(Q) \geq 0$ for all $Q \subseteq \mathbb{V} \times \mathcal{V}$. By definition, we have $H(\emptyset) = 0$. Consider an arbitrary non-empty $Q \subseteq \mathbb{V} \times \mathcal{V}$. For any $\gamma_i \in \mathcal{V}$ we have

$$0 \leq c_{\gamma_i} = \sum_{\substack{w \in \mathbb{V}: \\ (w, \gamma_i) \in Q}} F_w \leq \sum_{w \in \mathbb{V}} F_w = 1.$$

Therefore, $-c_{\gamma_i} \log c_{\gamma_i} \geq 0$ and

$$\sum_{i=1}^{C} L(c_{\gamma_i}) \geq 0.$$

Now we establish submodularity. Consider an arbitrary $Q \subseteq \mathbb{V} \times \mathcal{V}$, $R \subset Q$ and any $(w', \gamma_{i'}) \notin Q$. Let $Q' := Q \cup \{(w', \gamma_{i'})\}$, $R' := R \cup \{(w', \gamma_{i'})\}$. We need to show that

$$H(R') - H(R) \geq H(Q') - H(Q). \quad (4)$$

Let us denote the frequency of the unigram γ_j in Q, Q' as $c_{\gamma_j}(Q)$, $c_{\gamma_j}(Q')$. Since Q and Q' differ only in the group $\gamma_{i'}$ we have

$$H(Q') - H(Q) = \\ - c_{\gamma_{i'}}(Q') \log c_{\gamma_{i'}}(Q) + c_{\gamma_{i'}}(Q) \log c_{\gamma_{i'}}(Q) \quad (5)$$

Similarly, (5) holds for $H(R') - H(R)$. Thus, to proof (4) it is enough to show

$$-c_{\gamma_{i'}}(R') \log c_{\gamma_{i'}}(R') + c_{\gamma_{i'}}(R) \log c_{\gamma_{i'}}(R) \geq \\ -c_{\gamma_{i'}}(Q') \log c_{\gamma_{i'}}(Q') + c_{\gamma_{i'}}(Q) \log c_{\gamma_{i'}}(Q)$$

We have $c_{\gamma'_i}(Q') = c_{\gamma'_i}(Q) + F_{w'}$; therefore, (5) can be rewritten as $L_{F_{w'}}(c_{\gamma_{i'}}(Q))$. Similarly, $c_{\gamma'_i}(R') = c_{\gamma'_i}(R) + F_{w'}$ hence we need to establish

$$L_{F_{w'}}(c_{\gamma_{i'}}(R)) \geq L_{F_{w'}}(c_{\gamma_{i'}} Q). \quad (6)$$

For any $(w, i') \in R$ we have $(w, i') \in Q$; thus $c_{\gamma_{i'}}(R) < c_{\gamma_{i'}}(Q)$, and (6) follows from the fact that $L_{F_{w'}}(x)$ is monotone decreasing for all non-negative real x. □

Proof of Theorem 1. By the result (Lee et al., 2009), the Algorithm 5 outputs the map γ' such that

$$\frac{1}{4 + 4\varepsilon} H(\gamma^*) \leq H(\gamma'). \quad (7)$$

where γ^* is the grouping which achieves largest value of H. We need to show that the approximation guarantee still holds if $\gamma'(w)$ is undefined for some w.

After Step 8, the groupings γ' and γ differ only for the group i_0; thus,

$$H(\gamma) - H(\gamma') = L\left(c_{\gamma_{i_0}}\right) - L\left(c_{\gamma'_{i_0}}\right).$$

Assume that $H(\gamma) - H(\gamma') < 0$. First, there must exist $j \in \mathcal{V}$ such that

$$L\left(c_{\gamma'_{j_0}}\right) \leq \frac{1}{C} H(\gamma')$$

and thus for the group i_0 we have

$$L\left(c_{\gamma'_{i_0}}\right) \leq \frac{1}{C} H(\gamma') \quad (8)$$

From (8) and $L(x) \geq 0$ we obtain

$$L\left(c_{\gamma_{i_0}}\right) - L\left(c_{\gamma'_{i_0}}\right) \geq -L\left(c_{\gamma'_{i_0}}\right) \geq -\frac{1}{C} H(\gamma')$$

hence

$$H(\gamma) \geq \frac{C-1}{C} H(\gamma') \geq \frac{C-1}{4C + 4\varepsilon C} H(\gamma^*).$$

For a single matroid constrain, the algorithm from (Lee et al., 2009) runs in time $(|\Omega|)^{O(1)}$ where Ω is the universe. In our case, $\Omega = \mathbb{V} \times \mathcal{V}$ hence the running time is $O(C|\mathbb{V}|)^{O(1)}$. The rest of the Algorithm 5 takes $O(C|\mathbb{V}|)^{O(1)}$ steps, thus we obtain the stated running time and finish the proof. □