Using External Knowledge to Improve Brown Clustering

Veljko Miljanic

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science

University of Washington

2020

Committee:

Gina-Anne Levow

Fei Xia

Program Authorized to Offer Degree:

Linguistics

University of Washington

Abstract

Using External Knowledge to Improve Brown Clustering

Veljko Miljanic

Chair of the Supervisory Committee:
Gina-Anne Levow
Linguistics Department

In recent years, semi-supervised learning methods that rely on using low-dimensional word representation gained interest in NLP, due to their ability take advantage of vastly available unlabeled data and reduce dependence on large, labeled datasets. In this thesis we propose two methods which allow integration of domain specific knowledge to one of the most popular methods for low dimensional word representations – Brown clustering. First, we propose changing the order in which words are clustered so that words more relevant for the task are given priority. Then, we also propose modifying the clustering objective so that it ensures relevance of induced clusters for downstream supervised task. Experiments show that both methods improve performance of NER system using cluster features.

TABLE OF CONTENTS

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1. INTRODUCTION

Since the late 20th century, most Natural Language Processing research has focused heavily on machine learning models. Interest for this approach was driven by Moore's law which dictated increase in computation power and at the same time increased availability of large text corpora. Since beginning of the machine learning revolution in NLP until today, the most successful approaches are based on supervised machine learning algorithms. These algorithms use statistical models that are inferred from, usually large quantities, of labeled training data.

The conventional supervised approach in NLP is based on lexicalized features such as bag of words (BOW). In this model, the set of words that show up in a document is represented by a vector whose components each correspond to one unique word from the training corpora. For all words that appear in the document, we set corresponding feature to 1 and all others to 0. Besides other deficiencies, such as disregard of word order and grammar, BOW and similar feature representations suffer from data sparsity problem. This is because the feature vector has size of the vocabulary which is a very large number. Also, BOW is incapable of capturing similarity between words. Therefore, supervised models trained with BOW features have issues with inference on words that are rare in training corpora and they can't handle words that are not in training corpora.

With increased availability of large unlabeled corpora in the internet age, one popular approach to address the data sparsity issue is to use unsupervised methods to induce low dimensional word representations. These representations are designed to drastically reduce the number of dimensions by grouping semantically similar words together. One common approach to induce unsupervised

word representations is to use clustering and then base the feature vector on cluster identity instead of word identity. If induced clusters capture semantics well, the downstream supervised model will be able to generalize inference learned on frequent words to rare words that share the same cluster. Thus, models that leverage such low dimensional representations can achieve higher accuracy for the same amount of labeled data or same accuracy for smaller labeled data set.

One popular clustering method for induction of unsupervised word representations is Brown clustering (Brown *et al.*, 1992). This is hard, hierarchical word clustering algorithm that induces binary cluster hierarchy based on distributional similarity between words. Originally algorithm was developed for purposes of language modeling to induce class-based language model. However, in recent years it's frequently used in many NLP applications to address problem of data sparsity and improve performance on words which are not present in the training set.

As with all other unsupervised word representations, Brown clusters are generated without any knowledge about the task in which they will be used. Therefore, success of unsupervised word representations depends solely on how well the method for inducing them naturally fits with the domain in which they will be applied. In this thesis, we will explore if the quality of word representations can be improved by using knowledge from the target domain. We will present an approach which modifies the Brown clustering algorithm and introduces constraints based on quality of the cluster features that it produces. New word representations can no longer be considered as unsupervised. Rather they are semi-supervised and are tuned for use in a target application. We will use the Named Entity Recognition task to show that new semi-supervised

word representations outperform unsupervised representations induced by the classic Brown's algorithm.

To summarize, the contributions of this thesis are as follows:

1. Propose use of external knowledge to adapt Brown clustering for the domain in which cluster features will be applied.
2. Present two methods for incorporating external knowledge into clustering algorithm.
3. Show that the adapted cluster word representations outperform standard Brown cluster representation on Named Entity Recognition task.

Chapter 2. LITERATURE SURVEY

The Brown clustering algorithm (Brown et al., 1992) comes from the statistical language modeling field. It was introduced as a method for automatic induction of class-based language models for use in speech recognition and machine translation tasks. They presented a bottom up, hard, hierarchical clustering algorithm, that is assigning words to classes based on the frequency of their co-occurrence with other words. The algorithm starts by assigning each word to its own class and proceeds in bottom up fashion by merging classes based on log-probability of text under a class bigram language model. The authors showed that 3-gram class-based language model achieves slightly lower perplexity but reduces the model size three times. What is more relevant for this work is that they also observed that induced clusters capture semantics or syntactic similarity to

some extent. It was exactly this property of the clustering algorithm which has driven its applications in many Natural Language Processing tasks.

A similar clustering method, based on the exchange algorithm was presented by Kneser and Ney (Kneser and Ney, 1993) which also used class bigram language model perplexity as optimization criteria but with exchange clustering algorithm. They showed that the resulting class-based language model has batter quality than a regular word n-gram model. They also observed that their algorithm produces classes of syntactically similar words. Later Martin et al. (Martin, Liermann and Ney, 1998) extended this approach to cluster trigrams and evaluated induced language model in speech recognition system.

However, due to the property of the Brown algorithm to group syntactically and semantically similar words into clusters, the method later became very popular in supervised learning tasks. Miller et al. (Miller, Guinness and Zamanian, 2004) have pioneered the idea that word clusters can be used for feature generation in supervised learning. They used the hierarchical cluster tree to derive word representations which are then used to generate features for a Conditional Random Fields the NER tagger (explained in detail in section: Cluster features). Their results show that using cluster features reduces the amount of labeled data needed to train NER tagger. This turned out to be a very effective method to enhance performance of supervised systems by leveraging large amounts of unlabeled data, and it quickly found application in many NLP tasks. For example Liang (Liang, 2005) have applied cluster features in Chinese word segmentation task and Named Entity Recognition, Owoputi on Part of Speech Tagging task of

tweets and IRC conversations (Owoputi et al., 2012) (Owoputi et al., 2013) and Koo on dependency parsing (Koo, Carreras and Collins, 2008).

Because the number of clusters is many times smaller than number of words in the corpora, the Brown cluster word representations are low dimensional and help with data sparsity issue in NLP. The idea to use large amounts of unlabeled data to induce low dimensional word features for use in supervised tasks gained popularity and many methods were developed for this purpose. Turian et al. (Turian et al., 2010) did a comparative study of different word representations in Brown clusters to neural word embeddings (Collobert and Weston, 2008) and scalable hierarchical distributed language model (HLBL) embeddings (Mnih and Hinton, 2008) on NER and syntactic chunking tasks. In this study Brown clusters have outperformed other two methods on both tasks.

There has been a little research into improving the quality of Brown clusters. Derczynski et al. investigated impact of initial cluster count hyper parameter and corpus size (Derczynski, Chester and Bøgh, 2015). To measure the quality of resulting clustering they used part of speech tagging and named entity recognition tasks. Their work shows that cluster quality increases with input corpora size as well as increase in number of word classes. Also, their work shows that performance tends to plateau and then start decreasing when the number of clusters becomes too large or the corpus becomes too big. In later work, Derczynski and Chester looked into improving clustering algorithm (Derczynski and Chester, 2016). The modified algorithm allows them to run clustering once with a large merge window size and then produce word clusters in a postprocessing "rollup" step for the desired number of clusters C. They also propose a new method for extracting cluster features which makes more use of clustering quality (average mutual information).

In this thesis we will also investigate modifying the Brown clustering algorithm to improve clustering quality. However, in this work we take inspiration from constrained clustering algorithms. This is a subfield of semi-supervised learning that is studying how domain knowledge, such as must-link and can't link constraints or class statistics, can be included into clustering methods. For example, (Wagstaff et al., 2001) developed a K-means clustering algorithm that can incorporate external knowledge via must-link and can't link constraints. They found that even using a small amount of constraints can lead to significant improvement of clustering quality. Constrained clustering has been applied to many well-known clustering methods such as K-means, mixture modeling, and density based clustering (Ian Davidson, Sugato Basu, 2008).

In the constrained clustering literature, external knowledge is usually expressed in the form of must-link and can't link constraints. Such constraints are not available in NLP problems where Brown clustering is applied. Therefore, in this work we will consider using soft constraints inspired by feature selection methods in NLP. Comprehensive study of feature selection methods used in NLP is given by (Yang and Pedersen, 1997).

THE BROWN ALGORITHM AND ITS APPLICATIONS

2.1 NOTATION

n	Total number of words in input corpora.
$w_1, \ldots, w_n$	Input word corpora: w_i represents i-th word in corpora.
$n(w)$	Total number of occurrences of word w in corpora.
$n(w, w')$	Total number of occurrences of word bigram w preceded by word w' in corpora
C	Clustering represented as function which maps words to clusters
$C(w_i)$	Cluster to which word w_i is assigned.
k	Total number of unique words in corpora.
$n(c, c')$	Total number of occurrences of class bigram (cluster c preceded by cluster c') in corpora.
$Quality(C)$	Clustering quality for clustering function C. This is the objective of the clustering algorithm.

2.2 THE BROWN ALGORITHM

Brown clustering (Brown *et al.*, 1992) is agglomerative hierarchical clustering that derives hierarchy of word clusters from unlabeled corpora. Table 2.1**Error! Reference source not found.** presents standard bottom-up agglomerative clustering algorithm. Clustering starts by initializing singleton clusters for each word. Then, in each main loop iteration, we merge two clusters that are selected so that clustering objective is maximized. Algorithm stops when all the clusters are merged into a single cluster which contains all the words. Output of the algorithm is a hierarchy of clusters which can be represented as a binary tree, in which leaves are words and nodes are clusters, generated by merging their child nodes.

Table 2.1 Standard bottom-up agglomerative clustering. C is current cluster map, Q is a clustering quality.

```
1 initialize C = {w_1, ..., w_N}
2 while |C| > 1
3     (a, b) = argmax_{a,b ∈ C} Q(C_aUb)
4     update C: merge clusters a and b
5 end while
```

Brown clustering is using criteria that are based on distributional similarity between words. For this reason, all words that share the same ancestor in the cluster hierarchy will have some level of distributional similarity. The deeper the cluster is in the hierarchy the higher is the distributional similarity between words its subtree contains.

2.2.1 *Clustering objective function*

Brown clustering is using an objective function that is defined in the context of a class-based bigram language model. In this model each word belongs to exactly one class and the probability of a word sequence is estimated from class bigram probabilities and probabilities of particular words being generated by corresponding clusters.

Given clustering C which maps words to clusters, clustering quality $Quality(C)$ is calculated as logarithm of probability of word sequence $w_1, \dots, w_n$ (input corpora) in class-based bigram language mode normalized by number of words:

$$Quality(C) = \frac{1}{n} logP(w_1, w_2, \dots w_n) \quad 2.1$$

$$= \frac{1}{n} logP\big(w_1, w_2, \dots w_n, C(w_1), C(w_2), \dots, C(w_n)\big) \quad 2.2$$

$$= \frac{1}{n} log \prod_{i=1}^{n} P\big(C(w_i)\big|C(w_{i-1})\big)P\big(w_i\big|C(w_i)\big) \quad 2.3$$

Here $C(w_i)$ represents the cluster to which the i-th word in sequence is assigned, $P(C(w_i)|C(w_{i-1})$ represents the cluster bigram probability of cluster $C(w_i)$ following cluster $C(w_{i-1})$, $P(w_i|C(w_i))$ represents probability of word w_i being generated by cluster $C(w_i)$. We also assume that $C(w_0)$ is a special start cluster.

Equation (3.3) is not convenient for computation of clustering objective as it requires us to scan through the entire word sequence $(w_1, w_2, \dots, w_n)$ and therefore has linear complexity with respect to number of words in the corpora. However, we can transform it so that it only depends on number of cluster bigrams and number of word unigrams.

$$Quality(C) = \frac{1}{n} log \prod_{i=1}^{n} P\big(C(w_i)\big|C(w_{i-1})\big)P\big(w_i\big|C(w_i)\big) \quad 2.4$$

$$= \frac{1}{n} \sum_{w,w'} \frac{n(w,w')}{n} logP\big(C(w')\big|C(w)\big)P\big(w'\big|C(w')\big) \quad 2.5$$

$$= \frac{1}{n} \sum_{w,w'} \frac{n(w,w')}{n} \log \frac{n\big(C(w),C(w')\big)}{n\big(C(w)\big)} \frac{n(w')}{n\big(C(w')\big)} \quad 2.6$$

$$= \sum_{w,w'} \frac{n(w,w')}{n} \log \frac{n\big(C(w),C(w')\big)n}{n\big(C(w)\big)n\big(C(w')\big)} + \sum_{w,w'} \frac{n(w,w')}{n} \log \frac{n(w')}{n} \quad 2.7$$

$$= \sum_{c,c'} \frac{n(c,c')}{n} \log \frac{n(c,c')n}{n(c)n(c')} + \sum_{w'} \frac{n(w')}{n} \log \frac{n(w')}{n} \quad 2.8$$

Where w' represents next word and w represents current word; $c' = C(w')$ represent following cluster and $c = C(w)$ represents current cluster; $n(w)$, $n(c)$, $n(w', w)$ and $n(c', c)$ represent counts of word unigrams, cluster unigrams, word bigrams and cluster bigrams in input text respectively.

Now we can replace cluster and word counts with their probabilities:

$$Quality(C) = \sum_{c,c'} P(c,c') \log \frac{P(c,c')}{P(c)P(c')} + \sum_{w'} P(w') \log P(w') \quad (2.9)$$

$$= I(C) - H \quad (2.10)$$

Term $I(C)$ represents mutual information between adjacent clusters and term H represents entropy of the word distribution. Note that H is constant with respect to clustering C and therefore can be left out of the equation.

2.2.2 *Incremental computation of quality*

The clustering quality equation (2.9) has complexity of $O(k^2)$ where k represents number of clusters. For bottom up clustering approach (Table 2.1) we consider k^2 possible cluster pairs in each iteration. Given that in each iteration, the number of clusters is being reduced by one we have k iterations in total. Therefore, the complexity of clustering algorithm is $O(k^5)$. Given that the number of unique words in large unlabeled corpora is measured in millions this is still too high to be computationally tractable. However, (Brown *et al.*, 1992) showed that computation complexity can be further reduced to $O(k^3)$. This is achieved by maintaining a table L which maps each cluster pair to change in clustering quality that their merge would cause.

Let's examine cluster quality delta $L(c_a, c_b)$ after merging clusters c_a and c_b. We denote the new cluster as $c_a \cup c_b$, the current set of clusters as C and the set of clusters after merging c_a and c_b as C'. Where $C' = C - \{c_a, c_b\} + \{c_a \cup c_b\}$.

$$L(c_a, c_b) = Quality(C) - Quality(C') = \quad (2.11)$$

$$= \sum_{c,c' \in C} P(c,c') \log \frac{P(c,c')}{P(c)P(c')} - \sum_{c,c' \in C'} P(c,c') \log \frac{P(c,c')}{P(c)P(c')} \quad (2.12)$$

Note that clusters other than c_a and c_b, that are not considered for merging, are unaffected by the merge operation. This means that we can cancel out most of the terms in sums over cluster pairs in equation (2.12).

To simplify equations let's name the contribution of clusters c and c' to clustering quality as $v(c, c')$:

$$v(c,c') = \begin{cases} P(c,c')\log\frac{P(c,c')}{P(c)P(c')} + P(c',c)\log\frac{P(c',c)}{P(c')P(c)} & if\ c \neq c' \\ P(c,c)\log\frac{P(c,c)}{P(c)P(c)} & if\ c = c' \end{cases} \quad 2.13$$

Unless either c or c' is one of clusters affected by merge (c_a, c_b or $c_a \cup c_b$), $w(c, c')$ will be the same for both C and C' and will cancel out. This allows us to simplify $L(c_a, c_b)$ equation:

$$L(c_a, c_b) = \sum_{c \in C} \big(v(c_a, c) + v(c_b, c)\big) - \sum_{c \in C_i} v(c_a \cup c_b, c) \quad 2.14$$

Selecting clusters which produce the highest quality after merging is equivalent to finding clusters with maximum $L(c_a, c_b)$. Since the number of entries in table L is $O(k^2)$ we can compute entire table in $O(k^3)$ time.

However, it's still possible to reduce complexity by making incremental updates of L after each iteration. This is because v(c, c') will only change if:

1. either c or c' is one of the clusters that were merged in the previous iteration
2. either c or c' is a result of merging clusters in previous iteration.

Therefore, for each (c_a, c_b) where neither of them is created by merging clusters in previous iteration L can be updated in O(1) time:

$$\begin{aligned} \Delta L(c_a, c_b) = {} & v(c_a, c_k) + v\big(c_a, c_j\big) + v(c_b, c_k) \\ & + v\big(c_b, c_j\big) - v\big(c_a, c_k \cup c_j\big) - v\big(c_b, c_k \cup c_j\big) \end{aligned} \quad 2.15$$

Where c_k and c_j represent clusters that were merged in previous iteration.

Then for the rest of the cluster pairs, we calculate $L(c, c')$ using the equation (2.14). Note that for computation of v(c, c'), we need cluster unigram and bigram probabilities which we can obtain in O(1) time by making incremental updates after each iteration.

With these optimizations, complexity of maintaining table L is $O(k^2)$ and overall complexity of clustering algorithm is $O(k^3)$.

2.2.3 *Optimization using a fixed window size*

Still, given that k for reasonable corpora is at least a few hundreds of thousand words, the complexity of $O(k^3)$ is not computationally tractable and needs to be further reduced. This can be done by sorting words by frequency and placing W(window size) most frequent words in clusters. In each iteration, we would consider only pairs of words that are within the selected window for merge. After merge, the number of clusters becomes $W - 1$ and we can add create a new singleton cluster from the next most frequent word w_i into the window. The algorithm with fixed window optimization is given in Table 2.2.

Table 2.2 Brown clustering with fixed window size optimization. C is current cluster map, Q(C) is clustering quality (eq. 2.9), W is window size, L is table of quality delta (2.14)

```
1 Sort words by frequency: w_1, w_2, …, w_k
2 Initialize window C = {w_1, w_2, …, w_W}
3 Compute Q(C)
4 Compute L
5 i = W + 1
6 while |C| > 1:
7     (a, b) = argmax_{a,b ∈ C} L(a, b)
8     Q = Q + L(a, b)
9     Update L after merging a and b
10     Update C after merging a and b
11     if i < N:
12         Add word w_i to C
13         Update L after adding word w_i
14     i = i + 1
15 end while
```

Table L must be updated once new cluster c_i is added to the window. Update of table entries for clusters c_a and c_b where neither of them is newly added cluster c_i has complexity of $O(1)$:

$$\Delta L(c_a, c_b) = v(c_a, c_i) + v(c_b, c_i) - v(c_i, c_a \cup c_b) \qquad 2.16$$

For table entries, which involve added cluster c_i, we use equation (2.14) with complexity of $O(W)$. Therefore, complexity of updating all entries in L after adding a cluster is $O(W^2)$. Total complexity of the clustering with window optimization becomes $O(W^2 k)$. Note that this

complexity estimation doesn't include scanning through corpora to count word unigrams and bigrams and sorting of words per unigram frequency.

2.3 CLUSTER FEATURES

The result of the clustering algorithm is the cluster hierarchy, represented as a binary tree in which words are assigned to leaf nodes (clusters). The root node represents the cluster that contains the entire vocabulary. Interior nodes represent intermediate clusters that contain smaller clusters with similar distribution. Therefore, nodes lower in hierarchy represent clusters with a smaller number of words with higher correlation and nodes higher in the hierarchy represent clusters with larger number of less correlated words.

A common scheme for design of cluster features was pioneered by (Miller, Guinness and Zamanian, 2004). They assign a cluster name string to each word by traversing the path from root of binary cluster tree to its leaf and assigning 0 for each left branch and 1 for each right branch.

Table 2.3. Sample bit strings (Miller, Guinness and Zamanian, 2004)

lawyer	1000001101000	Nike	1011011100100101011100	John	101110010000000000
newspaperman	100000110100100	Maytag	101101110010010101111010	Consuelo	101110010000000001
stewardess	100000110100101	Generali	101101110010010101111011	Jeffrey	101110010000000010
toxicologist	10000011010011	Gap	10110111001001010111110	Kenneth	1011100100000000001100
slang	1000001101010	Enfield	101101110010010101111110	Phillip	10111001000000000011010
babysitter	100000110101100	genus	101101110010010101111111	WILLIAM	1011100100000000011011
conspirator	1000001101011010	Microsoft	10110111001001011000	Timothy	1011100100000000001110
womanizer	1000001101011011	Ventritex	10110111001001011010	Terrence	10111001000000000011110
mailman	10000011010111	Tractebel	101101110010010110011 0	Jerald	10111001000000000011111
salesman	100000110110000	Synopsys	101101110010010110011 1	Harold	101110010000000100
bookkeeper	1000001101100010	WordPerfect	101101110010010110100 0	Frederic	101110010000000101
troubleshooter	10000011011000110			Wendell	10111001000000011
bouncer	10000011011000111				

As shown in Table 2.3, words which are semantically similar tend to have common prefixes in cluster names, because they belong to same intermediate clusters. The longer the common prefix

is between two-word bit strings the higher is the level of similarity. This property can be used to generate features from bit string prefixes. To capture various levels of generalization we can use prefixes of different lengths. For example in (Miller, Guinness and Zamanian, 2004) prefixes of length 8, 12, 16, and 20 were used for previous, current and next word.

Chapter 3. USING EXTERNAL KNOWLEDGE TO IMPROVE BROWN CLUSTERING

Brown clustering is an unsupervised algorithm in which the solution depends only on word distributions in input corpora. Its objective is to find the cluster hierarchy which best fits class bigram language model objective. Being a traditional clustering algorithm, Brown clustering, only guarantees to group words according to their similarity and can't guarantee that clusters will provide optimal features for an underlaying NLP task. However, in real world applications we usually have information about the task in which cluster features will be used. Therefore, it's natural to explore if we can modify clustering algorithm to take advantage of this information and produce cluster features which better fit desired NLP task.

Here we will describe two modifications that improve the quality of cluster features by utilizing external knowledge from a labeled dataset. The training set for the supervised task in which cluster features are being induced can be used as a labeled set in clustering. The first proposal modifies the order in which words are considered for clustering. Instead of ordering words by their frequency in unsupervised corpora we propose using order based on importance of the word features in the labeled dataset. The second proposal introduces constraints based on the labeled dataset by modifying the clustering objective function. These constraints will prevent clustering decisions that would be damaging for quality of induced features and therefore guarantee that the resulting clusters will fit well with the domain in which they will be used.

Given that both labeled and unlabeled data are used to find clustering solution, this method can be classified as constrained clustering. Similar methods from the field of constrained clustering have been presented by (Ian Davidson, Sugato Basu, 2008). Just like the method that we will present here, this family of algorithms can incorporate domain knowledge into otherwise unsupervised clustering algorithms.

3.1 Using External Knowledge to Improve Word Order

As explained in section (2.1.3) the original Brown algorithm starts by placing each word into a singleton cluster and ordering them by word unigram frequency. Because of window optimization, the algorithm will only consider merging the first *w* clusters with the most frequent words. This logic is driven by the fact that the distribution estimates for most frequent words are the most accurate, and the cooccurrence matrix for these words will be less sparse.

In Brown's algorithm, a word is not considered in making a merge decision until the word is included in the cluster window. Once the word gets into clustering window, its co-occurrence will influence all subsequent clustering decisions. So, in other words, particular word co-occurrences will not impact clustering of any other word which was considered before it, but will impact clustering of all the words that are considered after it. Therefore, we can suspect that ordering of words plays a significant role on clustering outcome.

Ordering words by their frequency in unlabeled corpora intuitively makes sense if clusters are intended to be used in statistical language models. However, it's unclear if this approach is optimal if clusters are intended to generate features for supervised learning algorithm. For example, in Conll2003 corpora (Table 3.1) we see that all entity classes represent only 15% of overall words in the corpora. Although entity class words might be very significant to quality of clusters for NER system, they will be underrepresented in head words that have the most significant impact. To my knowledge, so far there was no comprehensive study of importance of word ordering on quality of clustering for supervised tasks.

Table 3.1 Entity class distribution in Conll2003 training set

Class	*Count*	*Ratio*
PER	11128	5%
LOC	8297	4%
ORG	10025	4%
MISC	4593	2%
O	188846	85%
Total	222889	100%

The more intuitive approach for supervised learning application would be ordering of words by their significance for a supervised learning task. In literature there are plenty of metrics used for feature selection which can provide such ordering. For example (Yang and Pedersen, 1997) gave an excellent comparison of the well-known metrics that are used in NLP. In this work we have experimented with ordering words by information gain.

The information gain measures bits of information obtained for a category by knowing the presence or absence of a feature in a document. The information gain for a feature x given a set of categories y_i where $i = \{1 \dots m\}$ is defined as:

$$IG = H(Y|X) - H(Y) \tag{3.1}$$

$$IG = -\sum_{i=1}^{m} P(y_i|x) log P(y_i|x) - \sum_{i=1}^{m} P(y_i|\overline{x}) log P(y_i|\overline{x}) + \sum_{i=1}^{m} P(y_i) log P(y_i) \tag{3.2}$$

The information gain metric given in equation (3.2) gives a measure of feature importance to classification task. However, we are looking for a word importance metric and, in NLP, it is common to generate multiple features for each term in the document. For example, in named entity recognition word identity features are generated separately for each position in a window around the word that is to be tagged. We solve this problem by computing the information gain value for each feature that word generates and then picking the maximum among them.

Table 3.2 Comparison of top 50 words: frequency word order vs information gain order

Word frequency order	Information gain order
the . , to of in and a " said on for 's that The is was by with be it as at from will not would has its have are he were an had which market but percent this been after government year their they his also company up	( ) . , the U.S. 's beat 3 in to - a Germany Britain of Australia and " England France 1 Cup on Russian Spain 2 0 New said AT German Italy South LONDON British President China Russia Japan Clinton Minister Pakistan United Open 4 Sweden 1. 2. 3.

As can be observed in Table 3.2 when words are sorted by decreasing information gain which will surface the most significant words for labeled dataset at the top: entity names, location prepositions, token ‘s indicating a possession and some symbols like brackets that for Reuters data are indication of named entity. In contrast frequency order does not have any named entity in top 50. As expected many of top 50 words in frequency order are common words which don't carry significant information for entity recognition: that, after, government, year, their, they, his…

Integration of information gain word order in the Brown clustering is fairly simple as it doesn’t require any modification of the core bottom-up clustering algorithm. One thing to consider is that information gain can be computed only for words which appear in training set. Therefore, in our implementation after all words from the training set have been added to the clusters, we process the rest of the words according to frequency word order.

3.2 CONSTRAINED CLUSTERING

We can view clustering as the process of induction of low dimensional word representations. At the beginning: we start with high dimensional word representations (singleton clusters). Then at each iteration we use an unsupervised objective function to merge two clusters and reduce dimensionality of word representations by one. However, since objective function is unsupervised it can cause merge decisions which are degrading the cluster feature quality for particular task. Therefore, it would be beneficial to modify the algorithm to prevent clustering decisions that will lead to poor cluster feature quality.

In this section we will propose a method which modifies the clustering objective function so that both unsupervised corpora and labeled training set are considered. This idea is illustrated in hypothetical example on Figure 3.1. For NER system with Person, Location and Organization categories we should prevent merging of Names cluster with other three sample clusters because it would cause degradation of cluster features for name and location categories. However, merging of Names, US States and Countries clusters would not have this negative effect because all of them represent location and therefore it should be allowed.

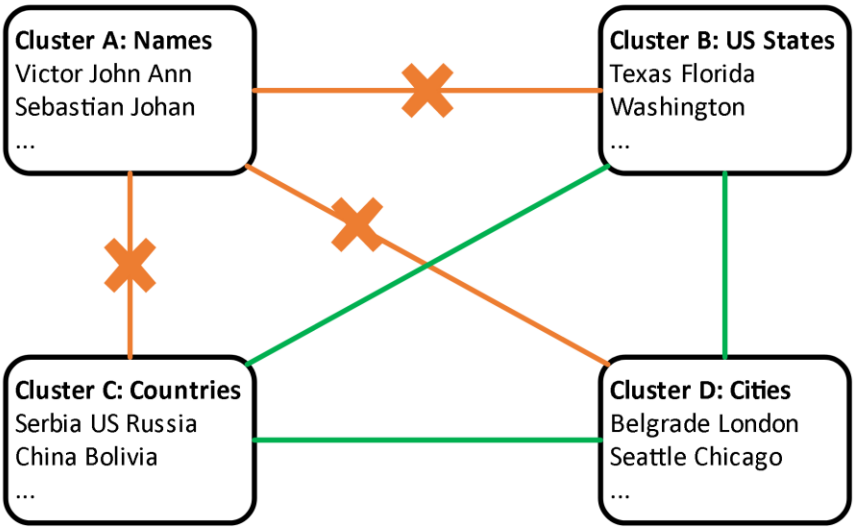

Figure 3.1 Constrained clustering example. Crossed over lines represent no-link constraints and clusters connected with them should not be merged.

To accomplish this, we will need a metric which can express the quality of cluster features for a downstream supervised task. There is variety of metrics used for feature selection (Yang and Pedersen, 1997) that can quantify importance of a feature to a labeled dataset without running a classification experiment. In this work, we will use feature information gain computed on the training set for the supervised task as measure of cluster quality. However, the proposed method should work with any metric which is simple enough to be computed on the fly during clustering.

3.2.1 *Feature Information Gain*

Information gain (IG) measures the number of bits of information obtained for a category preconditioned by knowing the presence or absence of a feature. For a binary feature $x \in \{0, 1\}$ information gain can be calculated as:

$$IG(x) = H(y) - H(y|x = 0) - H(y|x = 1) = \quad (3.3)$$

$$= -\sum_{i=1}^{m} P(y_i) log P(y_i)$$

$$+P(x = 0) \sum_{i=1}^{m} P(y_i|x = 0) log P(y_i|x = 0)$$

$$+P(x = 1) \sum_{i=1}^{m} P(y_i|x = 1) log P(y_i|x = 1) \quad (3.4)$$

Where x and y represent feature and label random variables, respectively, and m represents total number of labels.

3.2.2 *Information Gain for Pair of Clusters*

As explained in section 2.2 a bottom-up clustering algorithm will in each step merge a pair of clusters. Therefore, to integrate a supervised objective function into bottom-up clustering, we need to be able to compute the quality of clustering before and after merging two clusters. The merge operation will remove two clusters: a and b and add a new cluster aUb. In tagging tasks such as NER we will use cluster identity in word window to generate. Thus, each cluster will produce, not one, but a set of features. This makes feature information gain given by equation (3.2) insufficient for the task, since it can only estimate information after removing a single feature.

To address this issue, we will derive an information gain measure that reflects information gain after merging two clusters: a and b. In this work, we will assume that clusters will be used to produce features on word window of fixed size. Before merging clusters at each position on the windows we can have a value from set $\{a, b, \emptyset\}$ and after merge, each position can have value from

set $\{a \cup b, \emptyset\}$. Note that this problem is more complicated than estimating information gain defined by equation (3.2). That is because now we estimate information gain between two sets of features - $3n_w$ features before and $2n_w$ features after merging clusters a and b, where n_w represents window length.

Information gain after merging clusters can be computed as:

$$IG(c_a, c_b) = H\left(Y\middle|X_{\{a,b,\emptyset\}}\right) - H\left(Y\middle|X_{\{a\cup b,\emptyset\}}\right) \quad 3.5$$

$$= -\sum_{\overline{x}\in X_{\{a,b,\emptyset\}}} P(\overline{x}) \sum_{i=1}^{m} P(y_i|\overline{x}) log P(y_i|\overline{x})$$

$$+ \sum_{\overline{x}\in X_{\{a\cup b,\emptyset\}}} P(\overline{x}) \sum_{i=1}^{m} P(y_i|\overline{x}) log P(y_i|\overline{x}) \quad 3.6$$

Where $\overline{x}$ represents feature vector which contains cluster feature value at each position within a window (for example $\overline{x} = (x_{-2}, x_{-1}, x_0, x_1, x_2)$ for ±2 word window), $X_{\{a,b,\emptyset\}}$ and $X_{\{a\cup b,\emptyset\}}$ represent feature vector random variables before and after merging clusters a and b.

3.2.3 *Computing Feature Vector Probabilities*

Calculating information gain equation () is not as straightforward as it might appear. Problem comes from the fact that $\overline{x}$ is a vector of features in the window. Given that each feature can have 3 possible values before merge and 2 possible values after merge there are 3^5 or 2^5 of possible feature vectors per cluster for window of ±2 words. Therefore, number of possible values of $\overline{x}$ is too large to estimate $P(\overline{x})$ and $P(y_i|\overline{x})$ from $\overline{x}$ and y_i occurrence counts (equations 3.7 and 3.113.8) due to data sparsity issue.

$$P(\overline{x}) = \frac{count(\overline{x})}{N} \quad 3.7$$

$$P(y_i|\overline{x}) = \frac{count(y_i, \overline{x})}{count(\overline{x})} \quad 3.8$$

To address this problem, we will adapt the assumption that features generated at different positions within a window are independent. Therefore, we can estimate probabilities of P(x_i) and $P(x_i|y_j)$ at each window position i independently and then compute $P(\overline{x})$ and $P(y_i|\overline{x})$ from them:

$$P(\overline{x}) = \prod_{i=0}^{n_w} P(x_i) \qquad 3.9$$

$$P(y_j|\overline{x}) = \frac{P(\overline{x}|y_j)P(y_j)}{P(\overline{x})} = \frac{\prod_{i=0}^{n_w} P(x_i|y_j)}{\prod_{i=0}^{n_w} P(x_i)} \qquad 3.10$$

Now we can replace equations (3.9) and (3.10) into cluster merge information gain equation ().

Complexity of equations (3.9) and (3.10) depends linearly on feature window size and is $O(n_w)$. This gives us complexity of $O\left((2^{n_w} + 3^{n_w})mn_w\right)$ for equation (). However, m – number of classes and n_w – feature window size are constants and do not depend on sizes of supervised and unsupervised corpus. Therefore, with respect to corpus size complexity of computation is constant O(1).

3.2.4 *Constrained clustering algorithm*

The constrained clustering algorithm which combines both supervised (information gain) and unsupervised criteria is given in Table 3.3. In each iteration, the algorithm maintains table $IG(a, b)$ which stores the information gain if clusters a and b are to be merged. Instead, considering only unsupervised criteria $L(a, b)$ like the standard Brown algorithm given in Table 2.2, we will combine both supervised and unsupervised criteria with function $Combine(L(a, b), IG(a, b), i)$. Note that the index of next word i is provided to the function in order to turn off the supervised criteria when all words have been added and fall back to unsupervised criteria. There is no specific reason for this, except that in experiments we wanted to limit impact of supervision to formation of cluster tree nodes for easier analysis.

Table 3.3 Constrained Brown clustering algorithm. Difference with unconstrained clustering is in bold. C is current cluster map, Q(C) is clustering quality (eq. 2.9), W is window size, L is table of quality delta (eq. 2.14), IG is information gain table (eq. 3.5).

1 Initialize C = {c1, … cw}

2 Compute Q(C)

3 Compute L table

4 Compute IG table

5 i = W + 1

6 while |C| > 1:

7 $\mathbf{(a, b) = \underset{a,b \in C}{argmax}\, Combine(L(a, b), IG(a, b), i)}$

8 $Q = Q + L(a, b)$

9 Update L after merging a and b

10 Update IG after merging a and b

11 Update C after merging a and b

12 if i < k:

13 Add word w_i to C

14 Update L after adding word w_i

15 Update IG after adding word w_i

16 $i = i + 1$

17 end while

Computing updates of IG in steps 10 and 15 has complexity of $O(W^2)$. However, closer inspection of IG equation () shows that $IG(a, b)$ will remain constant unless either cluster a or cluster b changed (either result of merge in step 11 or added cluster in step 15). Also, if the changed cluster doesn't contain any word from the labeled corpora no change is necessary. Therefore, there is no

change in computation complexity and it remains the same as in the original Brown's algorithm: $O(w^2k)$.

3.2.5 *Combining Supervised and Unsupervised Cluster Quality*

At this point the only unsolved problem is how should unsupervised objective functions be combined. Unfortunately, these two represent two different metrics and are calculated on different datasets, so there is no natural way to combine them. Therefore, we have experimented with two empirical methods which nevertheless give solid results.

3.2.5.1 Linear scalarization

Linear scalarization is a standard method in multi-objective optimization. We can formulate a multi-objective optimization problem as single-objective by making linear combination of objectives. Then by varying coefficients of linear combination we can reach points in Pareto optimal solution. In our problem there are only two objectives, so we introduce parameter s_r (supervised ratio) which we can use to make their linear combination:

$$Combin(L(a,b), IG(a,b)) = s_r * IG(a,b) + (1 - s_r) * L(a,b) \qquad 3.11$$

s_r regulates how much supervised feature quality is important for the optimal solution. If s_r is equal to 1, we use only supervised objective and ignore original Brown clustering objective. For s_r equal to 0 the supervised objective is thrown away and the objective is identical to standard Brown clustering objective.

3.2.5.2 No-link threshold

In this method, we will use the feature quality objective in a similar manner in which information gain is used in feature selection to filter features. In every iteration of the clustering algorithm, we must choose a cluster pair to merge among $\frac{1}{2}W^2$ possible pairs. We use the supervised objective (IG) to rank all possible cluster pairs from highest to lowest. Then we declare all pairs ranked below a certain percentile threshold as no-link. The pair of clusters which will be merged is then selected from the rest of the cluster pairs by using only the unsupervised objective. The function $Combine$ for this method is given in equation 3.12:

$$Combine\big(L(a,b), IG(a,b)\big) = \begin{cases} -\infty & IG(a,b) < Pi\ threshold \\ L(a,b) & otherwise \end{cases} \quad 3.12$$

where $Pi\ threshold$ represents a value in IG table at desired percentile. For all cluster pairs which have IG lower than this value the function will return minus infinity therefore making them no-link.

The logic here is that we don't want the supervised objective to significantly interfere with which clusters to merge. This is because the supervised objective will favor merging words like: Lakers, USA, Microsoft, etc. into a single cluster as they are all indication of the ORG class. However, the ORG class consists of many subclasses such as companies, countries, sport clubs, non-government organizations, etc. By using this method, we still leave some room for the unsupervised objective to induce these sub-classes by their distributional similarity. This is important because distributional similarity is still the only way to make inference about rare words or words that do not appear in the training set.

Chapter 4. EXPERIMENTS

4.1 EXPERIMENT DESIGN

4.1.1 *Named Entity Recognition Task*

We have evaluated methods presented in Chapter 4 on the named entity recognition task (NER). In this task, the goal is to locate and classify named entities in unstructured text. Named-entities are strings that are names of people, places, organizations, etc. Named entity recognizers are widely used as a subcomponent in many NLP systems such as question answering, summarization, relation extraction, search, etc. A sample of NER tagger output is presented in Table 4.1.

Table 4.1. Sample NER tagger output

[PER **Novak Djokovic**] survived an injury scare and battled back from a set and a break down to help [ORG **Serbia**] take a 2-0 lead over [ORG **Russia**] in the [MISC **Davis Cup World Group**] tie on Friday.

The NER task is a task in which Brown cluster features are frequently applied. This makes it an ideal choice for evaluating performance of the proposed modifications to the Brown clustering algorithm.

4.1.2 *CoNLL-2003 NER dataset*

CoNLL-2003 dataset was made for language-independent named entity recognition task at the Conference for Natural Language Understanding 2003 (Sang and De Meulder, 2003). In this task named entities are defined as phrases that contain names of persons, organizations, locations and names of miscellaneous entities that do not belong to previous three categories.

Data for the task was taken from Reuters Corpus which consists of Reuters news stories between August 1996 and August 1997. For training and development sets, ten days of data were taken

from the end of August 1996. Test set data is from December 1996. This choice of split makes the blind set significantly more difficult because it is sampled from different timespan and is less likely to contain entities from the training set.

Table 4.2. CoNLL 2003 dataset statistics

	Articles	*Sentences*	*Tokens*
Training	946	14,987	203,621
Development	216	3,466	51,362
Test	231	3,686	46,435

Table 4.3. CoNLL 2003 entity statistics

	LOC	*MISC*	*ORIG*	*PER*
Training	7140	3438	6321	6600
Development	1837	922	1341	1842
Test	1668	702	1661	1617

4.1.3 *Named entity tagger design*

Named entity recognition is typically viewed as a sequence labeling problem. That is, given the observable (input) sequence $x = (x_1, \dots, x_N)$ the goal is to assign a categorical label to each member of input sequence $y = (y_1, \dots, y_N)$. Some of the most successful methods are statistical approaches where typical models include Hidden Markov Model (Rabiner and Rabiner, 1989), Conditional Random Fields (Lafferty, McCallum and Pereira, 2001) and sequential application of Perceptron or Winnow (Collins, 2002).

We based the tagger on Conditional Random Fields (Lafferty, McCallum and Pereira, 2001) machine learning model. This is a discriminative model which allows for usage of arbitrary

features and is a common approach in sequence labeling tasks. Distribution for linear chain CRF model that we used is modeled by the formula:

$$P(y|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} exp\left\{\sum_{j=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_j)\right\} \quad (4.1)$$

Where $Z(\boldsymbol{x}) = \sum_y \exp\left\{\lambda_y + \sum_{j=1}^{K} \lambda_{y,j} x_j\right\}$ is normalization constant, $\Lambda = \{\lambda_k\}$ are model parameters and $\{f_k(y, y', x_t)\}_{k=1}^{K}$ is set of real-valued feature functions.

In all experiments we have used CRFSuite package (Okazaki, 2007) which implements first order linear chain CRF training and decoding. To overcome limited context of the first order linear chain, we have transformed datasets so that expected labels are made by concatenating current and previous label. By doing this, we made an equivalent of second order CRF. In all experiments, we have used following the CRFSuite parameters for training: LBFGS algorithm with L1 regularization set to 0 and L2 regularization set to 0.0001. Each model was trained with 500 iterations through training set and we have evaluated the model produced by the last iteration.

Given that NER chunks can have multiple words while CRF assigns labels on the word level, it's necessary to use a chunk encoding scheme. we decided to use BILOU chunk coding as it was shown that it outperforms other common chunk coding schemes (Ratinov and Roth, 2009). This scheme suggests learning the classifier that identifies: **B**eginning, the **I**nside, the **L**ast tokens of multi token chunks as well as **O**utside tokens and **U**nit-length chunks.

As baseline features, we used features supplied with the NER sample in CRFSuite package. We added additional features for semi-supervised and unsupervised Brown clusters.

Table 4.4. Feature groups

Feature Group	Description
Custer	Brown cluster features in ±2 window
Words	Words in ±2 window
Word Type	Word type features in ±2. Word is assigned one or more types based on usage of uppercase, lowercase, digit and symbol characters. For example some of types are AllUpper, AllDigit, AllSymbols, etc.
Prefix/Suffix	Word prefix and suffix ±2 window. We use 3 and 4 characters for prefix and 1, 2, 3 and 4 characters for suffix
Word Shape	Word **S**hape in ±2 window. Word shape is generated by converting every character to character representing its type: **L** for lowercase, **U** for uppercase, **D** for digit, '**(**' for opening bracket, '**)**' for closing bracket, - for arithmetic sign, **;** for sentence end punctuation, **.** for dot or comma and **c** for everything else.
POS Tag	POS tags in ±2 window

Brown clusters were generated on Reuters Corpus Volume 1 (Lewis *et al.*, 2004) which contains Reuters News stories in English language. The corpus was preprocessed by running CoNLL 2003 task tokenizer and paragraphs with less than 80% lowercase characters removed as suggested in (Liang, 2005). Also, NLTK sentence tokenizer was used to break paragraphs on sentences and add <s> and </s> tags.

4.2 USING DOMAIN KNOWLEDGE TO IMPROVE WORD ORDER

In this section we will present results of experiments that evaluate the proposed approach for incorporating domain knowledge into word order for clustering.

4.2.1 *Evaluation on NER using only cluster features*

To evaluate information gain based word order described in section (3.1) we have trained NER systems which use only Brown clusters features for each of the word ordering methods:

- Random word order baseline
- Standard word frequency word order
- Information gain word order

Cluster features were generated by using cluster prefix method previously described in section (2.3). Following as (Miller, Guinness and Zamanian, 2004), we have used cluster prefixes of lengths: 4, 8, 12 and 16. Also we have tried each initialization method with different cluster counts: 100, 300 and 1000. Results of these experiments are presented in tables 4.5 and 4.6.

Table 4.5. Dev set: information gain vs word frequency initialization (only BC features)

	Random			Word Frequency			Information Gain		
Clusters	**P**	**R**	**f1**	**P**	**R**	**f1**	**P**	**R**	**f1**
100	69.38	63.19	66.14	65.26	59.91	62.47	**72.26**	**68.16**	**70.15**
300	72.87	71.07	71.96	70.29	67.87	69.06	**77.51**	**74.3**	**75.87**
1000	77.79	76.15	76.96	75.94	74.35	75.14	**80.59**	**78.63**	**79.6**

Table 4.6. Blind set: information gain vs word frequency initialization (only BC features)

	Random			Word Frequency			Information Gain		
Clusters	**P**	**R**	**f1**	**P**	**R**	**f1**	**P**	**R**	**f1**
100	64.44	60.45	62.38	61.23	56.07	58.54	**68.65**	**65.46**	**67.02**
300	68.68	67.6	68.14	65.26	64.29	64.77	**72.54**	**70.89**	**71.7**
1000	74.61	74.24	74.42	71.12	70.5	70.81	**76.4**	**75.55**	**75.97**

From Table 4.5 and Table 4.6 we can observe that information gain significantly outperforms word frequency initialization and random initialization in both precision and recall on both datasets. In the experiment with the highest cluster count of 1000 we gained 5.38 points on precision and 6.6 points on recall. Improvement in both precision and recall indicates that information gain ordering produces clusters that are both more accurate and have better generalization.

Perhaps, the most surprising result is that random word order performs better than word frequency order. The explanation for this is that named entities are frequent in Reuters corpora but there are very few specific named entities that show frequently enough to get into word window if frequency order is used. Therefore, random order seems to have a higher chance to select named entity words then a frequency word order. Detailed, analysis of this are presented in section 4.2.5.

Table 4.7 Summary of f1 improvements for information gain word order experiment

Clusters	*Dev f1*	*Blind f1*
100	7.68	8.48
300	6.81	6.93
1000	4.46	5.16

Information about f1 gain for each cluster count is summarized in Table 4.7. We can observe that as cluster counts increase the gain becomes lower. This can be explained by the fact that for higher cluster counts we have more chance to capture words that are relevant for NER task in initial cluster window. Also, this shows that using domain knowledge helps to "compress" relevant information better to smaller cluster count.

4.2.2 *Performance improvements on words in/out of training set*

One of the major benefits for using Brown clustering is that it improves performance on words which are not observed in the training set but share clusters with words that exist in the training set. Therefore, it's interesting to investigate whether the proposed method improves NER performance of words which are not in the training set and therefore had no supervision during clustering. To do this, we have divided words into two sets: **in-set** – words that are present in the training set and **out-set** – words that are not present in the training set (Table 4.8). Then we have computed performance metrics for both sets independently, to identify contribution of each group to overall gain.

Table 4.8 Statistics for in-set and out-set word groups.

	In count	*Out count*
Dev	47069	4293
Test	40779	5656

Table 4.9 Precision, recall and f1 for out-set word group measured on NER with cluster only features for cluster count of 1000.

	Development out-set			**Test out-set**		
	p	**r**	**f1**	**p**	**r**	**f1**
Freq. sort	74.12	68.45	71.17	73.81	70.75	72.25
IG sort	76.66	70.28	73.33	77.58	73.91	75.7
Delta	2.54	1.83	2.16	3.77	3.16	3.45

Table 4.10 Precision, recall and f1 for in-set word group measured on NER with cluster only features for cluster count of 1000.

	Development in-set			**Test in-set**		
	p	**r**	**f1**	**p**	**r**	**f1**
Freq. sort	80.97	79.82	80.39	76.63	76.47	76.55
IG sort	86.2	84.15	85.16	80.32	80.52	80.42
Delta	5.23	4.33	4.77	3.69	4.05	3.87

From Table 4.9 and Table 4.10 Table 4.10 it's clear that the proposed method yields significant gains in all metrics (precision, recall and f1) for both words that appear in the training set and words that do not appear in the training set.

4.2.3 *Evaluation on NER using all features*

Finally, we verify utility of the proposed ordering method on our best NER system which uses all features presented in the Table 4.4.

Table 4.11 Dev set information gain vs word frequency (all features)

	Word Frequency			Information Gain		
Clusters	**Precision**	**Recall**	**f1**	**Precision**	**Recall**	**f1**
100	89.75	88.98	89.36	**90.04**	**89.18**	**89.61**
300	90.31	89.36	89.83	**92.05**	**91.15**	**91.59**
1000	91.4	90.1	90.75	**92.99**	**91.7**	**92.34**

Table 4.12 Blind set information gain vs word frequency (all features)

	Word Frequency			Information Gain		
Clusters	**Precision**	**Recall**	**f1**	**Precision**	**Recall**	**f1**
100	84.31	83.36	83.83	**84.59**	**83.75**	**84.16**
300	84.58	83.82	84.2	**87.35**	**86.58**	**86.96**
1000	86.92	86.23	86.57	**87.99**	**87.66**	**87.8**

Table 4.13 f1 improvement on ner system with all features

Clusters	*Dev set f1*	*Test set f1*
100	0.25	0.33
300	1.76	2.76
1000	1.59	1.23

Results on the NER tagger with all feature groups again show improvements in both precision and recall metrics, for all feature counts. Still, in this experiment the gain is substantially smaller. For example, for cluster count of 1000 we obtained 1.23 f1 points on the test set, while with clusters only, the improvement was 5.16 f1 points. This can be explained by the fact that with addition of other features, performance of both NER systems significantly improves. Therefore, that some of improvements of information gain word order will overlap with improvements coming from other features.

4.2.4 *Impact of sorted word count vs NER performance*

Although sorting words by information gain word order will favor words which are of higher importance for NER tagging task, it might have a negative side effect, because some words with high information gain might have insufficient representation in unsupervised corpora for good word occurrence statistics. To analyze this, we did a set of experiments where only a limited

number of top words are ordered by information gain and the rest of the words are ordered by frequency.

Figure 4.1 shows NER f1 score dependency on number of top words that were selected by using information gain. Here we can observe that improvements do not saturate after the initial clustering word window is selected with information gain metric. The impact from sorting the rest of the words is significant and improves f1 for 3 points on development set and 2.4 points on test set. This is expected, as even though some words with high information gain do not end up in initial clusters they will be considered for clustering sooner and therefore have more significant impact on overall clustering result as discussed in section 0

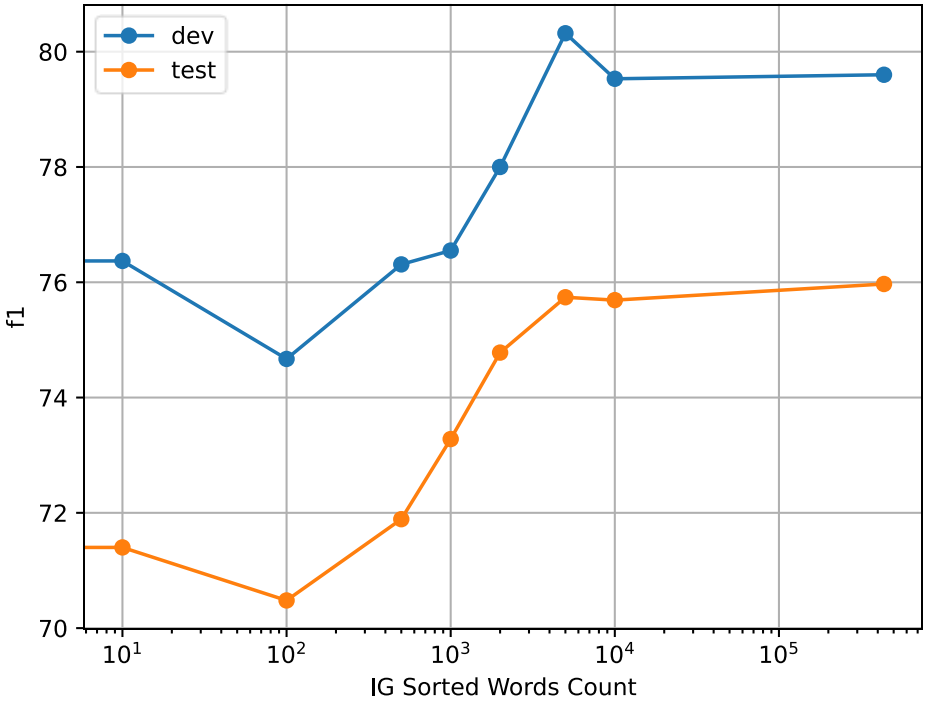

Figure 4.1 f1 score vs number of words sorted by information gain. Tested on NER with cluster only features (1000 window).

4.2.5 *Analysis*

The only difference between the regular brown clustering algorithm and the proposed solution is in the order in which words are considered for clustering. Top words for both standard Brown clusters (frequency order) and information gain order are presented in **Error! Reference source not found.**.

Table 4.14 Top 25 words comparison

Rank	*Random*	*Frequency*	*IG*
1	N.B.	the	(
2	critically-praised	.	)
3	airplane	,	.
4	permeating	to	,
5	Abrene	of	the
6	Anzag	in	U.S.
7	Semline	and	's
8	Smedervska	a	beat
9	levied	said	3
10	Graham-Parker	on	in
11	mulching	for	to
12	spending-related	's	-
13	Ukrainian-based	that	a
14	Ngurah	The	Germany
15	property-asset	is	Britain
16	uper-level	was	of
17	Jollas	by	Australia
18	Sensormatic	with	and
19	Penal	be	England
20	Longuet	it	France
21	otherwsie	as	1
22	off-the-track	at	Cup
23	Faridpur	from	on
24	non-State	will	Russian
25	Keramik	not	Spain

The most prominent difference is that in IG order we can observe many country names which obviously should help to form clusters relevant for location category. Also, word “Cup” is indication of sport events. Words “in” and “to” are indication of location. One might find surprising that brackets are the top IG words but that is due to Reuters articles header having “Author (Location)” format (e.g. “Peter Hedblom (Sweden)”).

Besides this we can observe that random word order got many named entities in top 25: Abrene, Anzag, Semline, Smedervska, Graham-Parker, Ngurah… The reason for this is that named entities are frequent in news articles and therefore random selection has a high chance to surface some named entities at the top of the word order. However, individual entities are rarely frequent enough, so they end up being suppressed at the lower ranks if word frequency order is used.

Table 4.15 Comparison of tagging results between NER trained only on cluster features: word frequency vs information gain word order.

Token	Expected	Frequency	IG
Chinese	**MISC**	**PER**	**MISC**
police	**O**	**PER**	**O**
have	O	O	O
detained	O	O	O
veteran	O	O	O
dissident	O	O	O
Wang	**PER**	**O**	**PER**
Donghai	**PER**	**O**	**PER**
,	O	O	O
the	O	O	O
New	**MISC**	**ORG**	**LOC**
York-based	**MISC**	**ORG**	**LOC**
pressure	O	O	O
group	O	O	O
Human	**ORG**	**O**	**ORG**
Rights	**ORG**	**O**	**ORG**
in	**ORG**	**O**	**O**
China	**ORG**	**LOC**	**LOC**
said	O	O	O
on	O	O	O
Saturday	O	O	O
.	O	O	O

Table 4.16 Clusters with frequency order represented as top 5 words with highest information gain

Human	**Rights**	**in**	**China**	**Said**
Human	Rights	in	China	Said
human	rights	water-gun	Japan	LAW
animal	spongiform	twixt	Pakistan	Malakwen
bovine	beings	topples	India	Claverie
yellow-brick	sex-and-murder	to-	Ukraine	Jemison

Table 4.17 Clusters with information gain ordering represented as top 5 words with highest information gain

Human	**Rights**	**in**	**China**	**said**
Human	Democracy	in	China	said
Financial	Peace	thatin	Pakistan	LAW
Business	Rights	tent-camp	Iraq	Malakwen
Natural	Reform	surrounding	Israel	Jemison
SoCal	Representatives	suggest	Ukraine	Nani

Comparison of NER results given in Table 4.15 gives an interesting insight on the advantage that clusters with information gain word order have over frequency word order. For example, phrase "Human Rights" is properly tagged as ORG if information gain word order is used.

The reason for this can be deduced from clusters corresponding to words Human and Rights that are given in Table 4.16 and

Table 4.17 Frequency word order places word Human (upper-case) into the same cluster as words: human (lower-case) and animal. However, the information gain cluster for Human contains words Business, Financial and Natural which are more likely to be part of organization name. Similarly, the word Rights is placed with uppercase words Democracy and Peace which are more likely to be in organization name then lowercase words rights and beings.

In this sample neither system was able to correctly classify the last two words of the phrase as organization. This is reasonable because the word "in" is very common. Also, the word China is recognized as location because it's more likely to be part of location name than organization. This

portion of the phrase would likely benefit from introduction of cluster-bigram features “Human in” and features based on previous tag: “ORG in”, “ORG China”.

4.3 CONSTRAINED CLUSTERING

In this section we will present experiment results with the constrained clustering approach described in section 3.2.

4.3.1 *Linear Scalarization*

To test constrained clustering with linear combination of supervised and unsupervised objectives (3.2.5.1), we run the constrained clustering algorithm with different s_r (supervised ratio) values in range $0 \leq s_r \leq 1$. All experiments are using cluster window of 1000 and entire CoNLL 2003 training set to compute the clustering supervised objective. Then we trained two NER systems: one with only cluster features and one with all features for each clustering result. F1 score for both systems is presented on Figure 4.2.

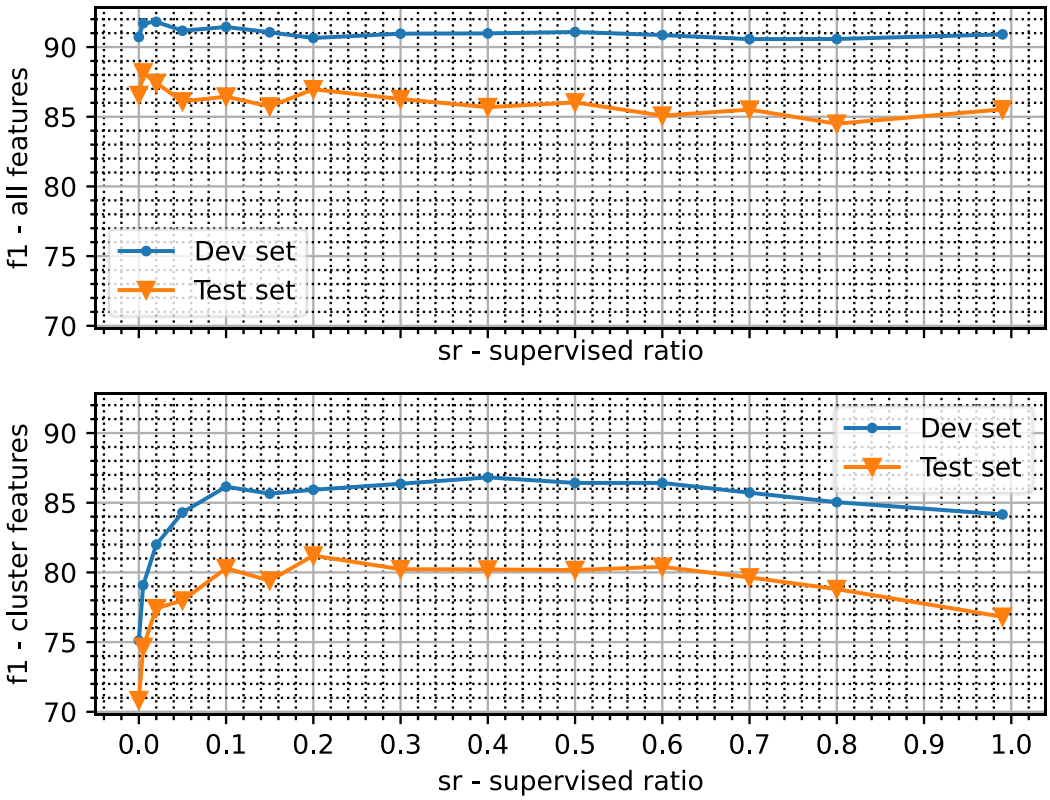

Figure 4.2: NER f1 score produced with constrained cluster features (linear scalarization of supervised and unsupervised objectives).

Results presented in Figure 4.2 show that constrained clustering substantially improved performance of NER system with only cluster features. This confirms our hypothesis that the proposed method to combine supervised and unsupervised objective functions will result in better cluster features. Also, the f1 score curve seems relatively smooth with respect to supervised ratio sr value allowing for easy selection of optimal value on development set.

However, NER system with all features shows entirely different result. F1 score plots are mainly flat and gains are measurable only for very small values of sr hyper parameter. Also, f1 score on test set is scoring worse for $sr > 0.05$ than the original Brown clustering algorithm ($sr = 0$).

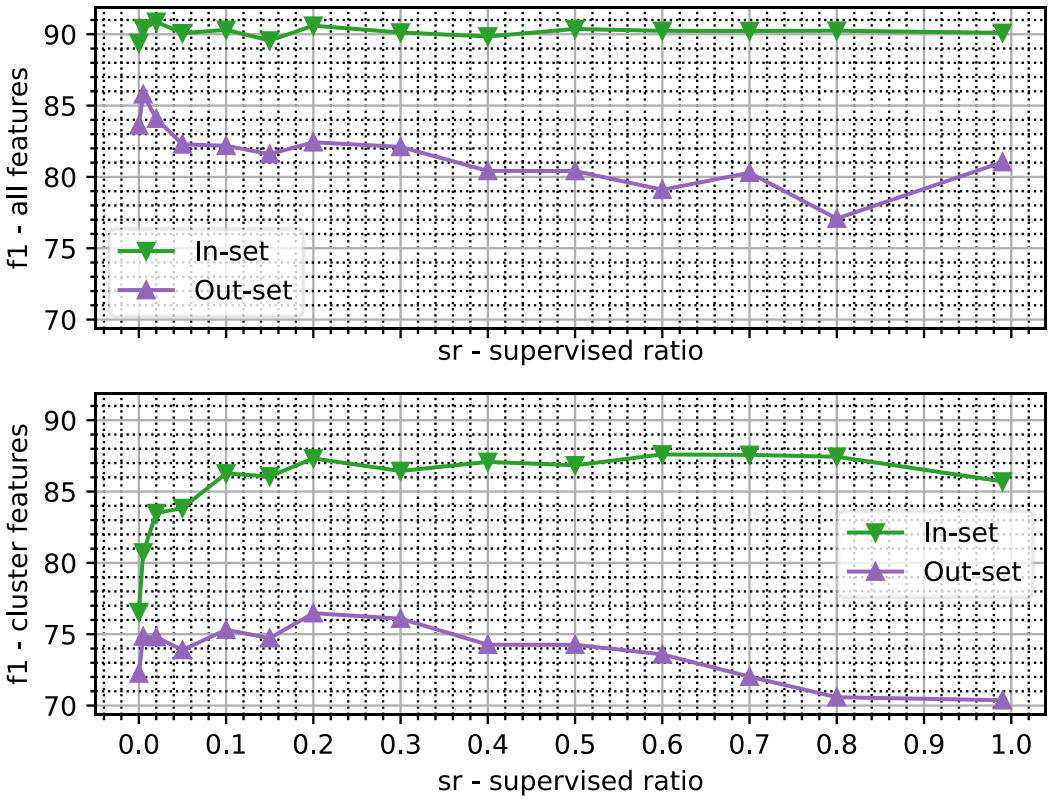

Figure 4.3 Comparison of in-set and out-set f1 score (test data) of NER system with all features.

These conflicting observations of NERs with only cluster features and NER with all features can be explained by difference in performance of constrained clusters on words observed in training set (in-set) and words only observed on test set (out-set) shown on Figure 4.3.

We can see that all improvements for NER with only cluster features are coming from words that are in-set. However, this is of little significance for NER with all features, because other features will allow system to learn how to handle words that appear in training set (in-set).

Out-set words showed modest improvement when used in isolation for $sr < 0.8$. However, when all features are used the improvement is only visible for very small values of $sr < 0.02$. One possible explanation is that for higher sr values clusters fit better in-set words, which could lead

NER to put too much confidence on cluster features as only words that can be observed during the training are in-set words. This would create problems on test data since it contains out-set words for which cluster features have much lower confidence.

4.3.2 *Linear Scalarization with Reduced Labeled data*

To address the issue of NER cluster features overfitting the training set, we have tried reducing amount of training data used for clustering. The portion of the words that were not included in clustering labeled dataset will become out-set words in NER training set and therefore NER should be able to learn how to handle them.

The subset for clustering was selected by sampling sentences from training set one by one until all in-set words from sampled sentences reached above 60% of development set. Then we added all sentences from training set which consist only of in-set words. The resulting set (reduced labeled set) is only 10% of original training data. However, this provides much more balanced in/out sets as shown in Table 4.18 and Table 4.19.

Table 4.18 In-set and out-set stats if entire training set is used for clustering.

Dataset	*Words*	*In-set*	*Out-set*
Train	203621	100%	0%
Dev	51362	92%	8%
Test	46435	88%	12%

Table 4.19 In-set and out-set stats if reduced labeled set is used for clustering.

Dataset	*Words*	*In-set*	*Out-set*
Train	203621	68%	32%
Dev	51362	63%	37%
Test	46435	63%	37%

Comparison of NER performance of clusters generated using whole training set and reduced training set is presented on Figure 4.4. The figure shows that clusters produced with whole training set significantly outperform clusters produced using reduced training set if used in NER system

with only cluster features. However, this is reversed if all features are used: reduced set clusters start outperforming clusters produced with whole training set.

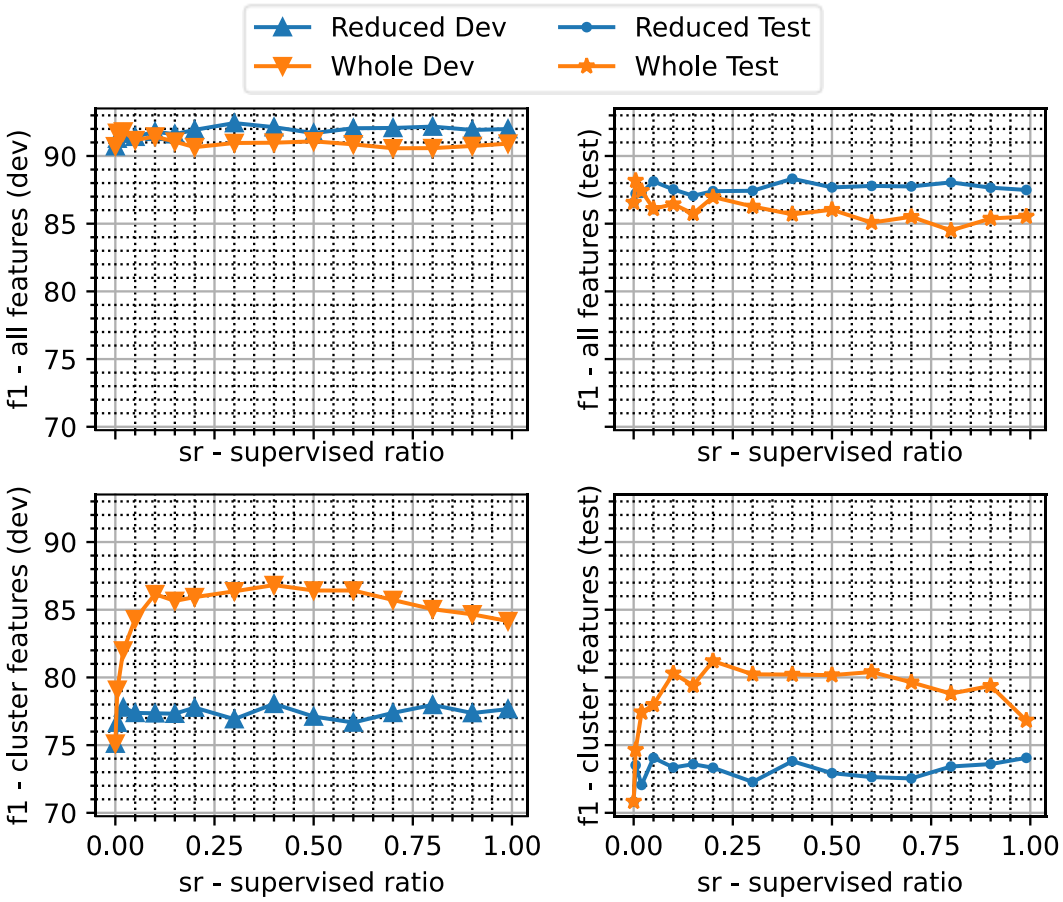

Figure 4.4 NER f1 score comparison for clusters generated with reduced labeled set and whole labeled (conll 2003 training) set.

Additional breakdown of f1 error on in-set words and out-set words (Figure 4.5) shows that using all training data for clustering improves performance on in-set words while regressing performance on out-set words. For real world cases in which cluster features are combined with other features, performance on out-set words is more important because in-set words can be handled well with other features.

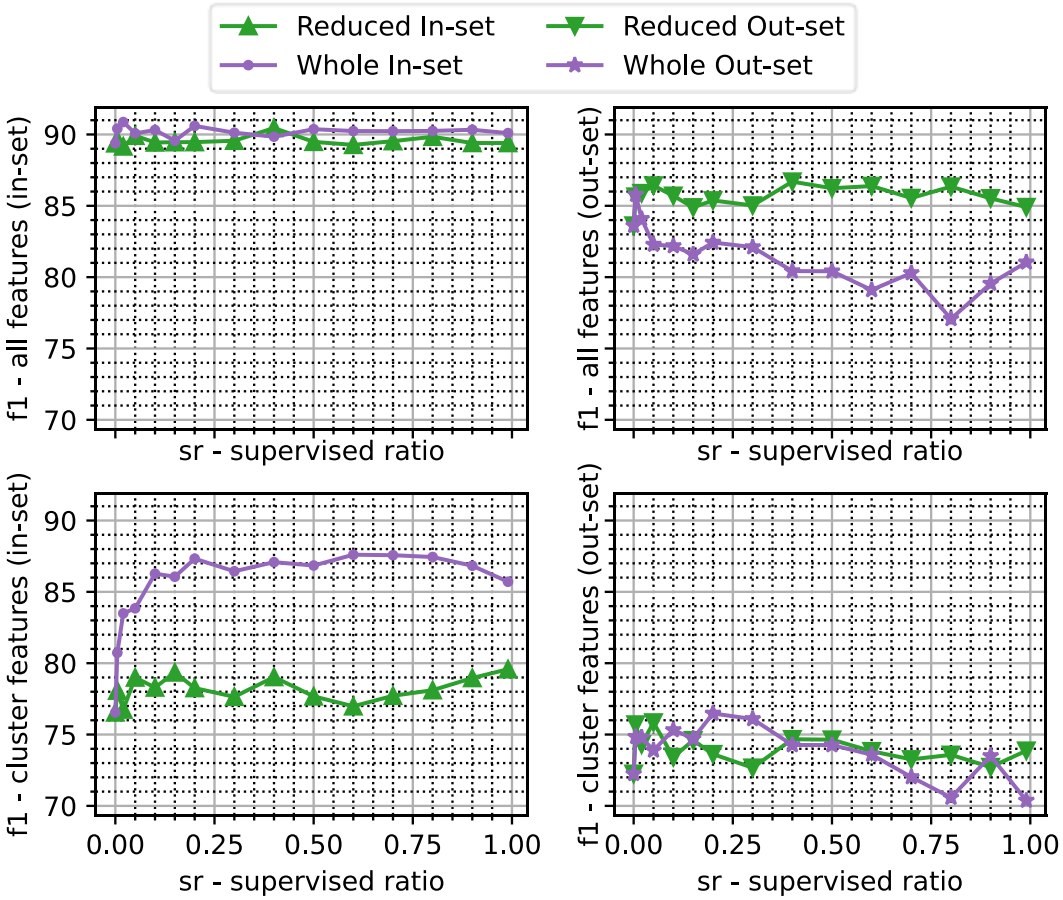

Figure 4.5 Comparison of in-set / out-set f1 score (test data) of clustering with reduced and whole labeled set.

4.3.3 *Supervised threshold experiments*

Results of clustering experiments of the constrained clustering method with no-link threshold (Section 3.2.5.2) are presented in Figure 4.6. Following the findings from experiment with linear scalarization and reduced labeled dataset (Section 4.3.2) we ran clustering with whole and reduced labeled sets. However, it turned out that for no-link threshold, reduced and whole dataset have similar performance when used in NER systems with all features.

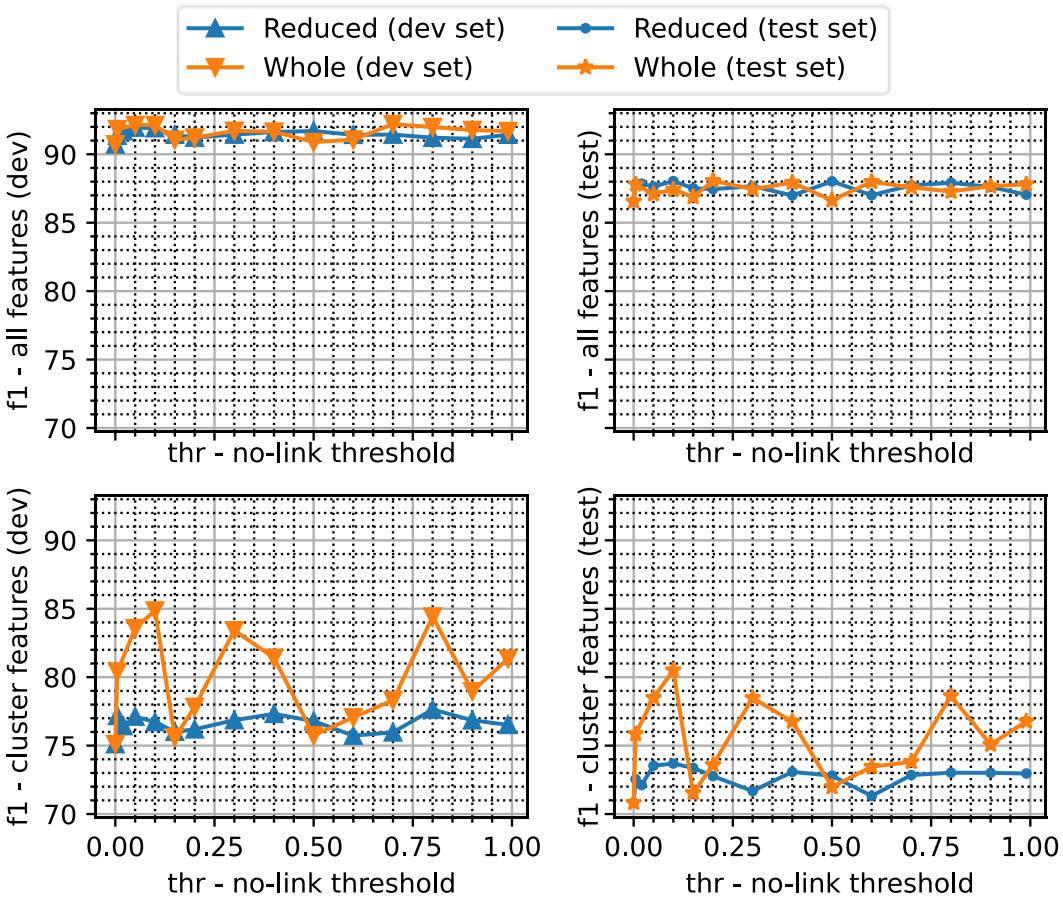

Figure 4.6 No-link threshold experiments. Reduced - clustering with reduced labeled dataset. Whole - clustering with entire labeled set.

4.4 SUMMARY OF RESULTS

A summary of the results discussed so far is presented in Table 4.20. For each proposed method we selected the hyperparameter (sr – supervised ratio or thr – no-link threshold) by scanning the range of possible values and selecting the value which gives the best f1 score on the development set. Also, we have tried to combine ordering of words with information gain with constrained clustering methods. In these experiments we performed the described procedure for selecting optimal constrained clustering hyperparameter since values from experiments without IG word order did not produce optimal results.

Table 4.20 Summary of all results. Agenda: liner - linear scalarization, no link – no link threshold, R – reduced labeled set clustering, W – whole labeled set clustering, IG sort – words sorted by information gain.

	Dev	Delta Dev	Test	Delta Test
Baseline	90.75	0	86.57	0
IG Sort	92.34	1.59	87.8	1.23
Linear W: sr=0.02	91.81	1.06	87.42	0.85
Linear R: sr = 0.3	**92.43**	**1.68**	87.43	0.86
No Link W: thr = 0.7	92.2	1.45	87.58	1.01
No Link R: thr = 0.1	91.91	1.16	88.04	1.47
IG sort + Linear R: sr = 0.01	92.25	1.5	88.07	1.5
IG sort + No Link W: thr = 0.5	92.25	1.5	87.74	1.17
IG sort + No Link R: thr = 0.2	92.38	1.63	**88.61**	**2.04**

Chapter 5. DISCUSSION

Using low-dimensional word representations derived from large unlabeled corpora has shown to be effective method to enhance performance of supervised NLP systems and reduce their dependency on large, labeled datasets. Brown clustering is one of the most popular methods due to its simplicity and competitive performance. In this thesis we have investigated whether Brown clustering can be improved by using external knowledge about supervised task during clustering.

We have proposed two different modifications of the clustering algorithm which allow integration of external knowledge in the form of labeled data. The first approach is to prioritize words by information gain on labeled dataset instead of frequency, therefore allowing words that carry more information to be considered first and form a seed for clusters. The second approach was to modify clustering objective so that both information gain on labeled dataset and unsupervised objective are considered. Our results summarized in Table 4.20 show that each of proposed methods is improving the performance of CRF based NER system.

The first proposed method - to replace frequency word order with word order of decreasing information gain on NER training set is very simple and doesn't require modification of the core clustering algorithm. We expected that the order in which words are clustered is important because the words that are clustered early form the seed of the clusters and therefore have higher significance. This is confirmed in experiments presented in section 4.2 in which using information gain word order improved performance of NER system for 1.6 f1 points on development set and 1.2 f1 points on test set. Further analysis showed that performance is improved even for words found only in test sets, therefore proving that selecting good seed words has positive impact for words which are considered much later in the clustering process.

We also proposed modifying clustering to consider both unsupervised and supervised optimization objectives combined with linear scalarization. The experiments in sections 4.3.1 and 4.3.2 showed that using all NER training data to compute the supervised objective can cause clusters to overfit the labeled training set. Since knowledge from the training set is easy to capture through other

features the proposed method ended up hurting NER performance except for very small, supervised objective mixing ratios. We successfully addressed this issue by reducing the amount of training data used in clustering. Our final results with this method showed 1.68 f1 score improvement on development set and 0.86 f1 score improvement on test set.

As an alternative to linear scalarization, we considered using a supervised objective to select pairs of clusters that should not be linked. This was showed to be more resilient to overfitting the training data and we observed similar improvements in experiments which used all and reduced labeled set in clustering. Overall improvement was on par with linear scalarization results.

Finally, we tried combining ordering words by information gain and two constrained clustering methods. Combining linear scalarization with information gain yielded improvement of 0.27 f1 points on test set and regression of 0.09 points on development set. Further, optimal mixing ratio in this experiment was only 0.01. This is much smaller than optimal ratio of 0.3 in test with linear scalarization alone, therefore suggesting that using both methods together can further elevate issues with overfitting the training data. Our best result was observed when using combination of information gain and clusters with no-link constrains on reduced dataset. This scored improvement, over IG word order, of 0.81 f1 points on test set and small 0.04 f1 improvement on development set. It's important to observe that constrained clustering methods are more complicated to implement then information gain word order and require tuning of additional hyperparameter. Yet, they are able to provide only a modest improvement in performance over simple information gain word order approach.

For scope of this thesis, we limited our experiments on CoNLL 2003 NER task which is a standard task for testing NER systems. However, our method is not limited to NER systems and we expect that the results should hold for all application where Brown clustering features have successfully been applied.

Chapter 6. CONCLUSION AND FUTURE WORK

The main hypothesis of this thesis is that Brown cluster word representations can be improved by modifying the clustering algorithm to consider fitness of the clusters for application in intended supervised learning task. We have considered changes to clustering word order and changes to clustering objective function which allow integration of knowledge derived from labeled training set for downstream supervised task. Experiments on Named Entity Recognition task showed that both proposed methods improve performance over standard Brown clustering algorithm and thus confirm our hypothesis.

The method presented in this thesis is not specific to the Brown clustering and it would be interesting to evaluate it with other clustering algorithms. Also, it would be interesting to investigate its possible application to neural word embeddings. Perhaps hierarchical cluster tree can be induced by using word embeddings to compute word similarity. This would allow simple integration of embedding based features in standard CRF taggers and also would allow application of the methods presented in this thesis.

BIBLIOGRAPHY

Brown, P. F. *et al.* (1992) 'Class-Based n-gram Models of Natural Language', *Computational Linquistics*, 18(1950), pp. 467–479.

Collins, M. (2002) 'Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms', *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1--8. doi: 10.3115/1118693.1118694.

Collobert, R. and Weston, J. (2008) 'A unified architecture for natural language processing', *Proceedings of the 25th international conference on Machine learning - ICML '08*, 20(1), pp. 160–167. doi: 10.1145/1390156.1390177.

Derczynski, L. and Chester, S. (2016) 'Generalized Brown Clustering and Roll-up Feature Generation', *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 1533–1539.

Derczynski, L., Chester, S. and Bøgh, K. S. (2015) 'Tune Your Brown Clustering, Please', *Proceedings of the conference on Recent Advances in Natural Lang Processing (RANLP)*, (2011), pp. 110–117.

Basu, S., Davidson, I., & Wagstaff, K. (Eds.). (2008). Constrained clustering: Advances in algorithms, theory, and applications. CRC Press.

Kneser, R. and Ney, H. (1993) 'Improved clustering techniques for class-based statistical language modelling', *Eurospeech*, 93(September), pp. 973–976.

Koo, T., Carreras, X. and Collins, M. (2008) 'Simple Semi-supervised Dependency Parsing', (June), pp. 595–603.

Lafferty, J., McCallum, A. and Pereira, F. C. N. (2001) 'Conditional random fields: Probabilistic models for segmenting and labeling sequence data', *ICML '01 Proceedings of the Eighteenth International Conference on Machine Learning*, 8(June), pp. 282–289. doi: 10.1038/nprot.2006.61.

Lewis, D. D. *et al.* (2004) 'RCV1: A New Benchmark Collection for Text Categorization Research', *Journal of Machine Learning Research*, 5, pp. 361–397. doi: 10.1145/122860.122861.

Liang, P. (2005) 'Semi-supervised learning for natural language', *Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science*, p. 86. Available at: http://hdl.handle.net/1721.1/33296.

Martin, S., Liermann, J. and Ney, H. (1998) 'Algorithms for bigram and trigram word clustering', *Speech Communication*, 24(1), pp. 19–37. doi: 10.1016/S0167-6393(97)00062-9.

Miller, S., Guinness, J. and Zamanian, a. (2004) 'Name Tagging with Word Clusters and Discriminative Training', *Proceedings of HLT-NAACL*, pp. 337–342.

Mnih, A. and Hinton, G. E. (2008) 'A Scalable Hierarchical Distributed Language Model.', *Advances in Neural Information Processing Systems*, pp. 1–8. Available at: http://discovery.ucl.ac.uk/63249/.

Okazaki, N. (2007) 'CRFsuite: a fast implementation of Conditional Random Fields (CRFs)', *http://www.chokkan.org/software/crfsuite*, p. 2007.

Owoputi, O. *et al.* (2012) 'Part-of-Speech Tagging for Twitter: Word Clusters and Other

Advances', *Cmu-Ml-12-107*, (September).

Owoputi, O. *et al.* (2013) 'Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters', *Proceedings of NAACL-HLT 2013*, (June), pp. 380–390.

Rabiner, L. R. and Rabiner, L. R. (1989) 'A tutorial on hidden Markov models and selected applications in speech recognition', *Proceedings of the IEEE*, pp. 257–286. doi: 10.1109/5.18626.

Ratinov, L. and Roth, D. (2009) 'Design challenges and misconceptions in named entity recognition', *Proceedings of the Thirteenth Conference on Computational Natural Language Learning CoNLL 09*, (June), p. 147. doi: 10.3115/1596374.1596399.

Sang, E. F. T. K. and De Meulder, F. (2003) 'Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition', *CONLL '03 Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, 4, pp. 142–147. doi: 10.3115/1119176.1119195.

Turian, J. *et al.* (2010) 'Word Representations: A Simple and General Method for Semi-supervised Learning', *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, (July), pp. 384–394. doi: 10.1.1.301.5840.

Wagstaff, K. *et al.* (2001) 'Constrained K-means Clustering with Background Knowledge', *International Conference on Machine Learning ICML*, pages, pp. 577–584. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.4624&rep=rep1&type=pdf.

Yang, Y. and Pedersen, J. O. (1997) 'A comparative study on feature selection in text categorization', *Machine Learning-International Workshop Then Conference-*, pp. 412–420. doi: 10.1093/bioinformatics/bth267.