Computing Word Classes Using Spectral Clustering

Effi Levi   **Saggy Herman**   **Ari Rappoport**
Institute of Computer Science, The Hebrew University
{efle|arir}@cs.huji.ac.il   saggy.herman@mail.huji.ac.il

Abstract

Clustering a lexicon of words is a well-studied problem in natural language processing (NLP). Word clusters are used to deal with sparse data in statistical language processing, as well as features for solving various NLP tasks (text categorization, question answering, named entity recognition and others).

Spectral clustering is a widely used technique in the field of image processing and speech recognition. However, it has scarcely been explored in the context of NLP; specifically, the method used in this work (Meila and Shi, 2001) has never been used to cluster a general word lexicon.

We apply spectral clustering to a lexicon of words, evaluating the resulting clusters by using them as features for solving two classical NLP tasks: semantic role labeling and dependency parsing. We compare performance with Brown clustering, a widely-used technique for word clustering, as well as with other clustering methods. We show that spectral clusters produce similar results to Brown clusters, and outperform other clustering methods. In addition, we quantify the overlap between spectral and Brown clusters, showing that each model captures some information which is uncaptured by the other.

1 Introduction

Word clusters (or *word classes*) are the result of dividing a lexicon of words into a pre-defined number of distinct groups, where the words comprising each group share some common characteristic. They have been well studied (Martin et al., 1998), in the context of part of speech (POS) induction (Christodoulopoulos et al., 2010), dealing with sparse data in statistical language processing (Brown et al., 1992), as well as for use as features in various NLP tasks such as text categorization (Bekkerman et al., 2003), question answering (Momtazi and Klakow, 2009), statistical parsing (Candito and Crabbé, 2009) named entity recognition (NER) (Miller et al., 2004), and others.

Brown Clustering (Brown et al., 1992), a hard hierarchical agglomerative clustering method, based on maximizing the quality of an induced class-based language model to make clustering decisions, is widely used to produce features for various NLP tasks (Momtazi and Klakow, 2009; Candito and Crabbé, 2009; Miller et al., 2004; Koo et al., 2008).

Spectral clustering is a clustering method which is broadly used for tasks such as image segmentation (Shi and Malik, 2000; Ng et al., 2002), speech recognition (Bach and Jordan, 2006) and topological mapping (Brunskill et al., 2007). It belongs to the family of dimensionality reduction algorithms and can actually be viewed as a weighted kernel K-means algorithm (Dhillon et al., 2004). However, it has scarcely been used for NLP tasks. Dhillon et al. (2011) use spectral methods for NER and chunking – but not for clustering – while Sun and Korhonen (2009) employ spectral clustering to improve verb clustering. Sedoc et al. (2017) used Signed Normalized Cut to produce word clusters; however, their motivation is focused on word-similarity (specifically, antonym-synonym relations), and they use mainly intrinsic evaluation methods. We aim to produce general spectral clusters and compare them to the widely-used Brown clusters using two NLP structured-prediction tasks for extrinsic evaluation.

While lacking the hierarchical nature of the Brown method, spectral clustering does possess a practical advantage over the former in that it produces – in addition to word clusters – a low-dimension represen-

tation $v \in \mathbb{R}^m$ for each of the words in the lexicon as well as for each of the cluster centers. Beyond obvious theoretical interest, this representation may be quite useful for measuring word-word, word-cluster and even cluster-cluster distances, as well as allowing us to perform soft clustering, in which we assign each word a distribution over the clusters (instead of assigning a single cluster).

In this work we use the spectral clustering method presented in (Meila and Shi, 2001) over a general lexicon to produce word clusters. We proceed to show that when used as features for solving two classical yet complex NLP tasks – SRL and dependency parsing – spectral clusters produce very similar results to those produced by Brown clusters, despite lacking the firm language-modeling grounding inherent in the latter. Finally, we quantify the overlap (and difference) in the information contained in the two cluster sets, and show that combining them could potentially induce significantly better performance than using each on its own.

2 Background

2.1 Clustering Methods

Clustering deals with the problem of dividing set of n samples into k distinct clusters following a desired function measuring sample similarity. The definition of what makes two samples similar or related is not obvious or singular; most intuitively, the samples may reside in $\mathbb{R}^n$ and the clusters should consist of samples that are similar following Euclidean distance, for example.

In the following sub-sections we will describe the best known clustering algorithm – *K-means* – which we use as a baseline, as well as spectral clustering – which is the focus of our work – and Brown clustering – which we compare to.

2.1.1 K-means

Dating back to six decades ago and published in (Lloyd, 1982), K-means is perhaps the most widely known clustering method. Given a set of points $s_1, ..., s_n$ and a set of k initial means $\mu_1^{(1)}, ..., \mu_k^{(1)}$ representing k clusters $C_1^{(1)}, ..., C_k^{(1)}$, the algorithm iterates until convergence:

1. For each point, find the cluster with the closest mean and add the point to that cluster:

   (a) $i = argmin_{C_j^{(t)}} ||s_p - \mu_j^{(t)}||^2$

   (b) $C_i^{(t)} \leftarrow s_p$

2. Update clusters means:

   $$\mu_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{s_p \in C_i^{(t)}} s_p$$

The means initialization may be random; different variants of K-means perform better with different types of initializations (Hamerly and Elkan, 2002).

K-means performs best when the data is spatially separable, and uses Euclidean distance as metric. It also has a tendency to choose clusters occupying area of similar size on feature space.

2.1.2 Spectral Clustering

Spectral clustering refers to a group of algorithms that work on the spectral (eigenvector) decomposition of the samples' *affinity matrix*, or *similarity matrix*. Each sample is represented by a vector; a metric is used in the resulting vector space to compute the distances between all point pairs – the affinity matrix. Following some mathematical manipulation on the affinity matrix, the eigenvectors are computed, and from them the clustering is derived, usually using the K-means algorithm. There are many flavors to this method, and we will describe here two of the main algorithms.

The first algorithm is introduced in (Ng et al., 2002):

Given a set of points $S = s_1, ..., s_n$ in $\mathbb{R}$ that we want to cluster into k subsets:

1. Form the affinity matrix $A \in \mathbb{R}^{n \times n}$ defined by $A_{ij} = \exp(-||s_i - s_j||^2/2\sigma^2)$ ($A_{ii} = 0$).

2. Define D to be a diagonal matrix s.t. $D_{ii} = \sum_j A_{ij}$, and $L = D^{-1/2}AD^{-1/2}$.

3. Find the k largest eigenvectors of L (matching the largest k eigenvalues), $x_1, \ldots, x_k$, and stack them in columns to form $X = [x_1 \ldots x_k] \in \mathbb{R}^{n \times k}$.

4. Derive Y from X by normalizing X's rows to have unit length.

5. Cluster Y's rows into k clusters (using K-means, for example).

6. Assign point s_i to cluster j iff row i of Y was assigned to cluster j.

The second algorithm is presented by Shi and Malik (2000; Meila and Shi (2001) and takes a graph-theoretic approach to clustering. The data set is represented by a weighted undirected graph. Each point is represented by a vertex, and each pair of vertices are connected by a weighted edge, with the weight representing the similarity between the two respective points. In this setting, they problem of segmenting the data set into two groups is formulated as partitioning the graph into two groups of vertices where the similarity within each group is maximized while the similarity between the groups is minimized.

Given a weighted graph $G(V, E, w)$ and two subsets of the vertices in the graph $A, B \subseteq V$, Define:

$$w(A, B) = \sum_{u \in A, v \in B} w(u, v),$$

The normalized cut, a symmetric measure for the disassociation between the subsets, is defined to be:

$$Ncut(A, B) = \frac{w(A, B)}{w(A, V)} + \frac{w(A, B)}{w(B, V)}$$

Along with a measure for association within each set:

$$Nassoc(A, B) = \frac{w(A, A)}{w(A, V)} + \frac{w(B, B)}{w(B, V)}.$$

Note that these two measures are related in the following way:

$$Ncut(A, B) = 2 - Nassoc(A, B)$$

Therefore, by minimizing the disassociation between groups we also maximize the association within each group.

Finding a cut that minimizes the *Ncut* criterion is shown to be NP-hard, and the *Ncut algorithm* is introduced, approximating in polynomial time the 2-way cut solution using the eigenvalues and eigenvectors of the affinity matrix. The algorithm is then used recursively to find a k-way partition of the graph, providing a clustering of the data set to k groups.

A more efficient method, however, is presented in (Meila and Shi, 2001). After establishing a probabilistic theoretic foundation to the normalized cut framework by offering a random walk interpretation, they present the *Modified Ncut algorithm* for a one-pass k-way segmentation:

1. Generate D as defined in (Ng et al., 2002) (the first algorithm).

2. Generate $P = D^{-1}S$, S being the similarity matrix, and find its eigenvalues and eigenvectors.

3. Discard the leading eigenvector and stack second-through-k leading eigenvectors to form $X = [x_2 \ldots x_k]$.

4. Perform K-means (or equivalent) on rows of X to find the clusters.

We chose to use the latter algorithm in our work, due to availability-of-code considerations.

Note that the last stage in the algorithm involves clustering the data points in the dense (due to the dimensionality reduction) vector space created by the computed eigenvectors. As mentioned in Section 1, this means that a by-product of this algorithm is a vector representation in this space for each of the clustered data points (in our case, lexicon words) as well as for the cluster centers. We discuss this further in Section 5.

2.1.3 Brown Clustering

Brown clustering (Brown et al., 1992) is a hard hierarchical agglomerative clustering method. It is based on the concept of maximizing the quality of an induced class-based language model, having originally been presented as a class-based n-gram model for natural language.

The algorithm generally follows the outline for hard hierarchical agglomerative clustering:

1. Start with a lexicon of types V

2. Sort V by corpus frequency, then put the top k types into clusters $C_1, ..., C_k$

3. Repeat $|V| - k$ times:
   - Put the next type into a new cluster C_{k+1}
   - Merge the pair in the $k + 1$ clusters to receive clustering C that maximizes $Quality(C)$

The function measuring the quality of the clustering at each iteration – $Quality(C)$ – was defined based on statistical language modeling reasoning. Given a corpus $w_1, ..., w_n$, a first-order Hidden Markov Model is used (with the clusters $C(w_1), ..., C(w_n)$ as the latent variables) to approximate the corpus probability:

$$P(w_1, .., w_n | C(w_1), ..., C(w_n)) \approx$$

$$\prod_{i=1}^{n} P(w_i | C(w_i)) P(C(w_i) | C(w_{i-1}))$$

The quality function is defined to be:

$$Quality(C) = \frac{1}{n} \log P(w_1, .., w_n | C(w_1), ..., C(w_n))$$

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

Further decomposing into:

$$= \sum_{1 \leq i,j \leq k} P(C_i, C_j) \log \frac{P(C_i, C_j)}{P(C_i) P(C_j)}$$

$$+ \sum_{w} P(w) \log P(w)$$

$$= I(C) - H(V)$$

Which is the mutual information of the clustering minus the entropy of the vocabulary. The latter is constant over the clustering, meaning that the mutual information is maximized over the clustering in each iteration of the algorithm.

2.2 Evaluation

Conducting a comparison between two different clustering methods is not a trivial task. Viewing the clusters themselves may provide us with some insight. Some criteria exist for assessing the quality of the clusters, such as silhouette graphs (Rousseeuw, 1987), while others, such as the Variation of Information criterion(VI) (Meilă, 2007) along with its normalized variant (Reichart and Rappoport, 2009), intend to compare two different clusterings over the same data set. These intrinsic methods, however, offer us very limited insight due to the unsupervised nature of our clustering task. We therefore turn to an extrinsic method of evaluation: using the clusters as features in a higher level task, and evaluate the clusters based on the performance in that task. We accomplish this by testing the clusters as features in two classical structured-prediction NLP tasks: semantic role labeling (SRL) and dependency parsing. As we wish to test the power of word clusters as features for these tasks, we use them exclusively without using any syntactic (POS tagging or dependency parses) or other semantic information.

2.2.1 Semantic Role Labeling

Semantic role labeling (SRL) is the task of detecting and labeling the different semantic arguments of a predicate in a sentence. The foundations for the task have been laid in (Gildea and Jurafsky, 2002), and had attracted much attention since. We use the PropBank annotated WSJ corpus, Palmer et al. (2005), which expands the Penn TreeBank with semantic annotations. These annotations were used as gold standard in the CoNLL-2005 shared task (Carreras and Màrquez, 2005).

While most often the task is broken down to two parts — finding the arguments using syntactic features, then labelling them using semantic features (Màrquez et al., 2008) – others try to solve the problem holistically, usually by training a statistical model such as Hidden Markov Model (HMM) or Conditional Random Fields (CRF), often solving some syntactic task jointly with SRL (Henderson et al., 2008). We adopt this approach in our work and use CRF for performing SRL.

Very few works attempt SRL without any syntactic features; Boxwell et al. (2011) report $F_1 = 0.44$ (although it is only reported for "completeness"). Recently, neural networks models have shown great promise in solving SRL. Boxwell et al. (2011), employing a unified neural network model to jointly learn POS tagging, chunking, NER and SRL, report $F_1 = 0.74$ on the SRL task. Zhou and Xu (2015), using a bi-directional long short-term memory (LSTM), report $F_1 = 81.07$, making their system state-of-the-art. These systems, however, are very complex and embody extensive fine-tuning in order to achieve the best possible results on SRL; our motivation, as previously stated, lies elsewhere.

2.2.2 Dependency Parsing

Dependency parsers build on the syntactic theory of *Dependency Grammar*. Proposed by Lucien Tesnière (1893-1954), the theory is based on relationships between words — between a *head* and a *dependent*. Starting from the verb, directed links connect all words of the sentence with links pointing from head to dependent, creating a directed rooted tree.

Dependency parsing is a major component of a large number of NLP applications. It is therefore one of the most well-studied tasks in NLP (McDonald et al., 2005; Nivre et al., 2007; Zhang and Nivre, 2011) and has been the focus of the CoNLL-2007 shared task (Nilsson et al., 2007). Extending classic syntactic features, Koo et al. (2008) use 4-6 bit prefixes of Brown clusters and full length clusters along with POS tags and word forms. They report an improvement in accuracy to 93.16%. over a baseline of 92.02%. Bansal et al. (2014) challenge Brown clusters by using word embeddings and performing hierarchical clustering on them. They reach accuracy of 92.69%, same as their Brown baseline, but with considerably faster training time.

3 Experimental Setup

3.1 Representation

in order to employ K-means over the lexicon as well as compute an affinity matrix for spectral clustering, we first need to decide on a representation for the words in our lexicon. We choose to use a simple window-based count model, in which each word is represented by the number of times it appears within a window (with a pre-defined size) around a pre-defined set of *descriptor words* in some corpus.

More formally, given a corpus, we choose the set of m most frequent words in the corpus to be our descriptor words, then each word w in the corpus is represented as a vector in $\mathbb{R}^M$, where each coordinate denotes the number of times w appears in the corpus in a window of a pre-determined size W on the **right** size of the respective descriptor word. The result is a $N \times M$ matrix (where N is the number of word in our lexicon) denoted R. We compute the exact same matrix using the **left** size of the descriptor word, and denote it L. These two matrices are finally concatenated to create a $N \times 2M$ sized matrix, in which each row contains the vector representation of a lexicon word, denoted the *context matrix C*.

In order to generate the context matrix we use the ukWaC corpus (Baroni et al., 2009), containing 2 billion words crawled from the `.uk` domain. Words are tokenized following the CoNLL format (separated `'s`, `'nt`, etc.), and the first word of a sentence is decapitalized.

For practical reasons, we artificially limit our lexicon size by choosing the top $N - 1$ most frequent words in the corpus, treating all other words as a special token "RARE" (making it the N-th word in our

lexicon).

For the experiments in this work we use $N = 12007$ and $M = 5000$ (the values were chosen empirically). In addition, we experiment with various values for the window size W (2, 3 and 5).

In addition, for completeness, we perform an additional set of experiments using a state-of-the-art word embedding as our lexicon words representation. We chose to use `word2vec`, implementing the continuous skip-gram algorithm presented in Mikolov et al. (2013a) with negative sampling (Mikolov et al., 2013b).

3.2 Clustering

3.2.1 Number of Clusters

We set the number of clusters on $k = 250$ throughout our experiments for all the clustering methods used in this work. This value was chosen empirically, having proven to provide the best results for both spectral and Brown clusters on the SRL task in preliminary experiments.

3.2.2 Affinity Matrix

As noted in Section 2.1.2, the spectral clustering algorithm required an affinity matrix, representing the affinity between each pair of samples (lexicon words) as input. Rather than using the default Gaussian kernel (Ng et al., 2002), we follow Sun and Korhonen (2009) using the *symmetrized skew-divergence* for generating the affinity matrix. This approach was found to produce better results during preliminary testing. Given two vectors v and v', their skew-divergence is

$$d_{skew}(v, v') = D_{KL}(v'||a \cdot v + (1 - a) \cdot v')$$

Where D_{KL} is the KL-divergence and v is smoothed with v' by parameter a (we empirically choose $a = 0.999$). The symmetrized skew-divergence is then defined as:

$$d_{s-skew}(v, v') = \frac{1}{2}(d_{skew}(v, v') + d_{skew}(v', v))$$

Finally, the affinity matrix is computed using this measure. Given the i-th and j-th lexicon words ($i \neq j$), we compute their respective vector representations v_i, v_j. We then set

$$A_{ij} = A_{ji} = d_{s-skew}(v, v')$$

For each i, we set $A_{ii} = 0$.

3.3 Software

3.3.1 Word Embeddings

In some of our experiments we utilize the `word2vec` embedding as word representation. We use the word embeddings available at `https://code.google.com/p/word2vec/`. These embeddings were produced by a network which was trained on a partial Google News data set ($\sim$100 billion words), and generated 300-dimensional vectors.

3.3.2 K-means

The K-means algorithm is employed as a baseline, as well as for the final stage in the spectral clustering algorithm (see Section 2.1.2). We use MATLAB's implementation of Lloyd's algorithm (Lloyd, 1982), with the K-MEANS++ algorithm for centroid initialization (Arthur and Vassilvitskii, 2007).

3.3.3 Spectral Clustering

We use the spectral clustering package presented in (Cour et al., 2004), implementing the NCut spectral segmentation algorithm presented in (Shi and Malik, 2000), available at `http://www.timotheecour.com/software/ncut/ncut.html`.

3.3.4 Brown Clustering

We use the implementation provided by Liang (2005), available at `https://github.com/percyliang/brown-cluster`. The clusters were computed on the ukWaC corpus described in Section 3.1.

3.3.5 CRF

For the purpose of solving SRL we wanted a simple yet powerful, off-the-shelf learning algorithm. As stated in Section 2.2.1, we choose to use CRF, feeling it is a simple yet powerful enough tool that would allow us to place the focus on the features rather than the learning process. We use the CRF++ package, available at `https://taku910.github.io/crfpp/`, implementing Lafferty et al. (2001)'s algorithm.

3.3.6 Dependency Parser

For our dependency parsing experiments we utilized the MSTParser, implementing the parser described in (McDonald et al., 2005) supplemented with second order features (McDonald and Pereira, 2006), available at `http://sourceforge.net/projects/mstparser/`.

4 Experiments & Results

4.1 SRL

Our first set of experiments is conducted on the PropBank annotated WSJ corpus (Palmer et al., 2005), which expands the Penn TreeBank with semantic annotations. We train a CRF model on sections 2-21 and test it on section 23. The features set for the CRF algorithm includes five surface features: whether the word contains a number, a hyphen or a capital letter, the position of the word w.r.t. current predicate and the word's length in characters. In each experiment we add one of the following sets of features:

1. K-means clusters (using simple count model decribed in 3.1)

2. Spectral clusters (using simple count model decribed in 3.1) [1]

3. Brown clusters

4. K-means clusters (using `word2vec`)

5. Spectral clusters (using `word2vec`)

6. POS tag

7. POS tag + dependency [2] parent's POS tag

8. POS tag + dependency [2] parent's POS tag + dependency [2] grandparent's POS tag

For each experiment we report per-argument precision, recall and F_1 score (Table 1).

Examining the first 3 rows in Table 1, we note the very small difference in F_1 score between the spectral model and the Brown model (0.003), denoting virtually equal performance. This is a very interesting result, considering that Brown clustering is tailored for word clustering by incorporating a statistical language model in the target function used for the clustering itself, as opposed to spectral clustering which takes no lexical considerations into account during the clustering process. Both models improve over the baseline (K-means) by approximately 0.04.

Moving down in the table, we see that experimenting with the `word2vec` embedding as initial representation yields worse results than using the simple count model. This is also a surprising result given the amount of success `word2vec` is having in word similarity tasks (Baroni et al., 2014). We suspect

---

[1] Using a combination of all three window sizes, discussed in Section 3.1

[2] Dependency parses were extracted from the CoNLL-2005 Shared Task annotated data set (Carreras and Màrquez, 2005).

Feature Set	precision	recall	F_1
K-means (count model)	0.620	0.504	0.555
Spectral (count model)	0.648	0.552	0.596
Brown	0.663	0.547	0.599
K-means (`word2vec`)	0.628	0.471	0.539
Spectral (`word2vec`)	0.642	0.503	0.565
POS	0.666	0.552	0.604
POS + parent's POS	0.661	0.556	0.604
POS + parent's POS + grandparent's POS	0.681	0.564	0.617

Table 1: Results for the SRL experiments.

Feature Set	UAS
Spectral (count model)	0.867
Brown	0.881
POS	0.921

Table 2: Results for the dependency parsing experiments.

this is due to the fact that these embeddings are learned without any connection to the clustering task, possibly not retaining some clustering-related information due to their high density.

Moving on, we observe that using POS tags as features instead of word clusters does not significantly improve the results: merely by 0.005 over Brown clusters. Considering the POS tag of the word's parent in the dependency tree does not improve the results, but considering the POS tag of the grandparent does improve performance by 0.013. All-in-all, we can improve over word clusters using syntactic information by the small amount of 0.018, but this requires second-order information from the dependency tree as well as POS tags, both which are generally expensive to manually produce.

4.2 Dependency Parsing

Our second set of experiments is conducted on the dependency annotations expansion to the WSJ corpus introduced in (Carreras and Màrquez, 2005). We train the MSTParser on sections 2-21 of the data set and test the resulting model on section 23. We use several of the feature sets described in Section 4.1 to compare between the different clustering techniques.

For each experiment we report unlabeled attachment score (UAS). Results are shown in Table 2.

We can observe that these results are similar to the ones obtained for SRL. The difference between using Brown clusters to spectral ones is relatively small, though larger than in the SRL task (0.014 out of 0.881). Using POS improves performance here too, though by a larger margin (0.041). Over all, we observe the same phenomena, albeit on a slightly different scale.

4.3 Oracle Model

In order to analyze the amount of information overlap between spectral and Brown clusters (in the context of SRL and dependency parsing performance), we examine the performance of an *oracle model*.

Given a prediction task, along with two models trained for the task, an oracle model is a hypothetical model which is capable of determining which one of the two models will perform better on any given test sample. The decision is made per sample; for each sample, the oracle chooses the better model. As stated, this is obviously a hypothetical model, but it gives us the opportunity to estimate how overlapping our two models are. In the case of a complete overlap, the performance of the oracle model will not improve beyond those of the separate models; if the overlap is not complete, however, the amount of improvement achieved by the oracle may indicate the amount of difference between the models.

We perform the following procedure for the models learned using the spectral vs. the Brown clusters (feature sets 2 & 3 in 4.1). We go over the test samples one by one and check which model performed better, taking that model's prediction to be the oracle's prediction on that test sample. Finally, we eval-

Task	Spectral	Brown	Oracle
SRL (F_1)	0.596	0.599	0.664
Dep. Parsing (UAS)	0.867	0.881	0.901

Table 3: Results for oracle model analysis.

uate the oracle model according to its predictions. We perform this analysis for both the SRL and the dependency parsing data sets.

Results – F_1 score for SRL and UAS for dependency parsing – are given in Table 3.

The oracle result for SRL reveals the complementary nature of the two methods, achieving a significant increase of 0.075-0.078 in F_1 score over using each model separately (an improvement of 12.5%) , outperforming even the best model which is based on syntactic features by 0.047 in F_1 score (compare to last row in Table 2). This interesting result means that the information captured by the spectral method may be quite different from the one captured by the Brown clusters.

A deeper examination reveals that while the two models agree on 61.4% of the sentences in the test set, the spectral model outperforms the Brown model on 18.8% of the sentences while the Brown model performs better on 19.8% of them, further emphasizing the complementary nature of the two clustering methods.

The oracle result for dependency parsing shows an improvement over the individual models as well, although a less dramatic one (an increase of 0.02-0.034 in UAS, which is a 2.3%-3.9% improvement). We hypothesize that the reason may be the syntactic nature of dependency parsing compared to the semantic nature of SRL, given that distributional word clusterings are usually assumed to capture mainly semantic information.

5 Conclusion

In this work we apply spectral clustering to produce word clusters for a general word lexicon. We compare them to a K-means baseline, as well as the acclaimed Brown clusters, by using the clusters resulting form each of these methods as feature for solving SRL and dependency parsing. For a complete comparison, we also examine the use of an advanced word embedding method (`word2vec`) as well as hand-crafted syntax-based features (POS tags, dependency trees) in the same setup.

Interestingly, we observe a very similar (virtually identical, in the case of SRL) performance by the spectral and the Brown clusters, both outperforming the other clustering methods, while being marginally outperformed by hand-crafted syntactic features. We find the similar performance exhibited by the spectral clusters, compared to Brown clusters, a very interesting result, given the linguistically-motivated nature of Brown clusters vs. the none-language-related spectral clustering method.

In our view, this result may serve to motivate the use of the spectral method for lexicon clustering. This motivation is further enhanced by an advantageous quality of this clustering method: as a by-product, it produces a low-dimension vector representation for each of the lexicon words as well as for each of the clusters themselves (see the end of Section 2.1.2). This representation may be used, for example, to produce more features, such as the i-th closest cluster for each word, or produce a *soft* clustering, assigning each word with the distribution over distances from the clusters. It may also be used it to remove word outliers from the clusters, or characterize the relations between the clusters (based on the distances between them). It may even be interesting to explore the properties of this representation as a word embedding for various NLP tasks.

An oracle analysis reveals that spectral clusters and Brown clusters complement each other rather than completely overlap. The oracle model achieves a significant improvement of 12.5% in performance in the SRL task, with a more modest improvement of 2.3%-3.9% in the dependency parsing task. We further show that the two models agree merely on 61.4% of the test samples in the SRL test set, approximately evenly dividing the rest of samples between them (in terms of besting each other). This analysis reveals the complementary nature of these clustering methods, implying that each of them may be better suited for different cases in the same task.

References

[Arthur and Vassilvitskii2007] David Arthur and Sergei Vassilvitskii. 2007. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics.

[Bach and Jordan2006] Francis R Bach and Michael I Jordan. 2006. Learning spectral clustering, with application to speech separation. *Journal of Machine Learning Research*, 7(Oct):1963–2001.

[Bansal et al.2014] Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *ACL (2)*, pages 809–815.

[Baroni et al.2009] Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.

[Baroni et al.2014] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Dont count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 238–247.

[Bekkerman et al.2003] Ron Bekkerman, Ran El-Yaniv, Naftali Tishby, and Yoad Winter. 2003. Distributional word clusters vs. words for text categorization. *Journal of Machine Learning Research*, 3(Mar):1183–1208.

[Boxwell et al.2011] Stephen A Boxwell, Chris Brew, Jason Baldridge, Dennis Mehay, and Sujith Ravi. 2011. Semantic role labeling without treebanks? In *IJCNLP*, pages 192–200.

[Brown et al.1992] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

[Brunskill et al.2007] Emma Brunskill, Thomas Kollar, and Nicholas Roy. 2007. Topological mapping using spectral clustering and classification. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3491–3496. IEEE.

[Candito and Crabbé2009] Marie Candito and Benoît Crabbé. 2009. Improving generative statistical parsing with semi-supervised word clustering. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 138–141. Association for Computational Linguistics.

[Carreras and Màrquez2005] Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 152–164. Association for Computational Linguistics.

[Christodoulopoulos et al.2010] Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two decades of unsupervised pos induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 575–584, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Cour et al.2004] Timothee Cour, Stella Yu, and Jianbo Shi. 2004. Normalized cut segmentation code. copyright 2004 university of pennsylvania. *Computer and Information Science Department*.

[Dhillon et al.2004] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. 2004. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556. ACM.

[Dhillon et al.2011] Paramveer Dhillon, Dean P Foster, and Lyle H Ungar. 2011. Multi-view learning of word embeddings via cca. In *Advances in Neural Information Processing Systems*, pages 199–207.

[Gildea and Jurafsky2002] Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

[Hamerly and Elkan2002] Greg Hamerly and Charles Elkan. 2002. Alternatives to the k-means algorithm that find better clusterings. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 600–607. ACM.

[Henderson et al.2008] James Henderson, Paola Merlo, Gabriele Musillo, and Ivan Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 178–182. Association for Computational Linguistics.

[Koo et al.2008] Terry Koo, Xavier Carreras Pérez, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *46th Annual Meeting of the Association for Computational Linguistics*, pages 595–603.

[Lafferty et al.2001] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Liang2005] Percy Liang. 2005. *Semi-supervised learning for natural language*. Ph.D. thesis, Massachusetts Institute of Technology.

[Lloyd1982] Stuart P Lloyd. 1982. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137.

[Màrquez et al.2008] Lluís Màrquez, Xavier Carreras, Kenneth C Litkowski, and Suzanne Stevenson. 2008. Semantic role labeling: an introduction to the special issue. *Computational linguistics*, 34(2):145–159.

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

[McDonald and Pereira2006] Ryan T McDonald and Fernando CN Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*.

[McDonald et al.2005] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics.

[Meila and Shi2001] Marina Meila and Jianbo Shi. 2001. A random walks view of spectral segmentation. *AI and STATISTICS (AISTATS) 2001*.

[Meilă2007] Marina Meilă. 2007. Comparing clusterings—an information based distance. *Journal of multivariate analysis*, 98(5):873–895.

[Mikolov et al.2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mikolov et al.2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Miller et al.2004] Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *HLT-NAACL*, volume 4, pages 337–342.

[Momtazi and Klakow2009] Saeedeh Momtazi and Dietrich Klakow. 2009. A word clustering approach for language model-based sentence retrieval in question answering systems. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1911–1914. ACM.

[Ng et al.2002] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. 2002. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856.

[Nilsson et al.2007] Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The conll 2007 shared task on dependency parsing. In *Proceedings of the CoNLL shared task session of EMNLP-CoNLL*, pages 915–932. sn.

[Nivre et al.2007] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

[Palmer et al.2005] Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

[Reichart and Rappoport2009] Roi Reichart and Ari Rappoport. 2009. The nvi clustering evaluation measure. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 165–173, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Rousseeuw1987] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

[Sedoc et al.2017] Joao Sedoc, Jean Gallier, Dean Foster, and Lyle Ungar. 2017. Semantic word clusters using signed spectral clustering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 939–949.

[Shi and Malik2000] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905.

[Sun and Korhonen2009] Lin Sun and Anna Korhonen. 2009. Improving verb clustering with automatically acquired selectional preferences. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 638–647. Association for Computational Linguistics.

[Zhang and Nivre2011] Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 188–193. Association for Computational Linguistics.

[Zhou and Xu2015] Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.