UNSUPERVISED TERM DISCOVERY FOR SIGN LANGUAGE

by

Korhan Polat

B.S., Electrical and Electronics Engineering, Boğaziçi University, 2017

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University
2020

ACKNOWLEDGEMENTS

First, I would like present my most sincere gratitude to Prof. Murat Saraçlar for his guidance. Without his invaluable insights and endless patience, this work would not be accomplished. I've always felt fortunate to be his student. I would like to thank Prof. Lale Akarun for her critical feedback and comments during this study. I would also like to thank Assist. Prof. Berk Gökberk for accepting to be in my defense jury and sharing his profound knowledge.

I am thankful to the friends and colleagues at BUSIM lab, namely Gözde Çetinkaya Ertop, Alican Gök, Öykü Deniz Köse, Enver Fakhan and Nazif Can Tamer, for their friendship and hospitality. There are no words that can express the debt of gratitude that I owe to Enver and Nazif, for the most insightful discussions we had and their friendship.

It was very fortunate for me to have weekly meetings with the Sign Language group and discuss our research. Therefore I am very grateful to Ahmet Alp Kındıroğlu, Gizem Esra Ünlü, Oğulcan Özdemir and Alptekin Orbay.

I am thankful to my friends, Can Bakışkan, Abdullah Karaaslanlı, Melih Dal, Miraç Sanisoğlu, Özge Bozal for inspiring me and Dr. Doruk Uludağ for making this journey a lot more enjoyable. I owe special thanks to Seden Gürlek for always motivating me with her stoic philosophy, Alper Taşkıran for the socially distanced companionship we had, during the online study sessions on each day of the quarantine period that lasted for months. At the end, I would like to thank my brother, mother and father for always being there for me, always believing in me and supporting me.

This work was supported in part by the Scientific and Technological Research Council of Turkey (TUBITAK) under Project 117E059.

ABSTRACT

UNSUPERVISED TERM DISCOVERY FOR SIGN LANGUAGE

Current sign language recognition systems rely on supervision for training successful models. However, in order to utilize the large amount of unlabeled sign language resources, unsupervised learning methods are needed. Motivated by the successful results of unsupervised term discovery in spoken languages, this work explores how to apply similar methods for sign terms discovery. The goal is to find the repeating terms from continuous sign videos without any supervision. Using visual features extracted from RGB videos, it is shown that discovery algorithms designed for speech can also discover sign terms. The experiments are run on a large scale continuous sign corpus and the performance is evaluated using gloss level annotations, for which time boundaries are given. The evaluation metrics are also inherited from spoken term discovery. This work unveils the potential use of unsupervised term discovery algorithms for continuous sign languages. .

ÖZET

İŞARET DİLİNDE GÖZETİMSİZ TERİM KEŞFİ

Gözetimsiz terim keşfi (GTK) son yıllarda konuşma işleme alanında *konuşulan terim keşfi* adıyla çalışılan bir problemdir. Amaç söz konusu dildeki tekrar eden sözcüklerin sadece konuşma sinyali kullanılarak keşfedilmesidir. Böylece kaynakların az olduğu dillerdeki terimler öğrenilmektedir. Etiketli veri kümesi sayısının az olduğu bir alan olan işaret dili işleme da gözetimsiz yöntemlere ihtiyaç vardır. Bu tezde, tekrar eden işaret terimlerinin gözetimsiz keşfi için konuşma işlemede uygulanan GTK algoritmaları işaret dili videolarına uyarlanmıştır. Ses öznitelikleri kullanmak yerine videolardan çıkarılan görsel öznitelikler kullanılmıştır. Konuşulan terim keşfinde kabul gören başarım ölçütlerinden faydalanarak, GTK yöntemlerinin işaret terimi keşfindeki başarısı ölçülmüştür. Elde edilen sonuçlara göre, GTK yöntemi işaret diline uygulandığında, konuşulan terim keşfindeki uygulamasına benzer başarım göstermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

a	Interval of segmentation for KNN based method
$cos(x, y)$	Cosine distance between two vectors x, y
d_x	Length of μ-percentile filter
d_y	Length of Gaussian smoothing kernel
g_L	Triangular function
$h(x)$	Bit signature of vector x
k	Number of neighbours for KNN search
l_i	Length of i^{th} segment in a pair
m	Start index of a segment
n	End index of a segment
s_p^1	The first segment belonging to p^{th} pair
s_t	HMM state at time t
w_1^N	Word sequence of N words
x_1^T	Feature vectors from time 1 to T
B	Beam width of neighbour search for E-SDTW method
D	Diagonal probe length for SDTW search
$D_\phi(\mathcal{X}, \mathcal{Y})$	Warping distortion between two sequences vectors
E	Set of edges
G	Adjacency graph
$H(x, y)$	Hamming distance between two bit signatures
$F_{olap}(s_k, s_l)$	Fraction of overlap between two segments
L	Length of a segment
L_{min}	Length constraint for LCMA
M	Similarity or distance matrix
M'	Binarized similarity matrix
$\mathcal{N}$	Normal distribution
$\mathcal{P}$	Set of matching segment pairs
P	Number of sorting permutations

R	Width of the Segmental DTW band
V	Set of nodes or vertices
γ	Scaling ratio for fusion
δ	Similarity threshold for E-SDTW method
ζ	Ratio of matches to keep for KNN based method
θ	Cost threshold
μ	Threshold for μ-percentile filter
μ_p	Alignment cost for segment pair p
ρ	Threshold for Hough transform
$\tau(s_i)$	Gloss transcription of a segment
ϕ	DTW warp path

LIST OF ACRONYMS/ABBREVIATIONS

2D	Two Dimensional
ASLR	Automatic Sign Language Recognition
BLSTM	Bidirectional Long-short Term Memory
CAE	Correspondence Auto Encoder
CNN	Convolutional Neural Network
DGS	German Sign Language, (Deutsche Gebärdensprache)
DH	Deep Hand
DTW	Dynamic Time Warping
DPGMM	Dirichlet Process Gaussian Mixture Model
EM	Expectation Maximization
E-SDTW	Efficient Segmental Dynamic Time Warping
GMM	Gaussian Mixture Model
GPU	Graphical Processing Unit
HMM	Hidden Markov Model
KNN	K-Nearest Neighbour
LCMA	Length Constrained Minimum Average
LSH	Locality Sensitive Hashing
MFCC	Mel-Frequency Cepstral Coefficients
NED	Normalized Edit Distance
NMS	Non-maximal Suppression
OE	Overlap Edges
OP	Open Pose
PCA	Principal Component Analysis
PLEB	Point Location in Equal Balls
RGB	Red Green Blue
SC	Segment Center
SDTW	Segmental Dynamic Time Warping
UTD	Unsupervised Term Discovery

VAE	Variational Auto Encoder
ZR	Zero Resource

1. INTRODUCTION

Sign languages are the most important tool of communication for the deaf or mute people, whereas they are foreign languages for most of the society. This brings a communication gap between the deaf and the rest of the world. In addition, some sign languages are used only by small village communities, which makes them more difficult to study due to lack of language resources. In order to close this communication gap, sign language researchers have been making use of computational tools such as automatic sign language recognition (ASLR) systems.

Most of the modern ASLR systems rely on supervision and linguistic expertise for training. Supervision is provided with the annotation of manual features (hand-shape, location), non-manual features (facial expression) or gloss information. With the recent advancements in computer vision and deep learning, ASLR systems are becoming more complex and the demand for annotated data is increasing even more. Even though there exist plenty of sign language recordings, most of them are not annotated since manual annotation is a labor intensive task and has to be carried out by linguistic experts. The lack of annotated corpora hinders the development of ASLR systems and it is one of the reasons that explains the slow accomplishments in this field compared to other computer vision fields. However, there exist plenty of sign language video datasets comprised of broadcast TV news, movie interpretations, conversations, where some of them are accompanied with weak text or speech labels, whereas some do not have any kind of labels at all. In order to make use of these unlabeled datasets, there arise the need for language independent, unsupervised learning procedures. With this goal set, it is worth investigating and get inspiration from the unsupervised learning methods in spoken language processing, since both sign language and speech share common properties.

Unsupervised learning has been an active research area in speech processing. Motivated by the fact that infants learn to recognize words without access to orthographic

labels, speech researchers aimed to develop models that learn subword representations and word-like units from speech, from the speech signal itself. The extreme case, where there is neither knowledge of linguistic structure nor labeled training data, is referred as the *zero resource* setting [2, 3]. Zero resource speech processing research focuses on two main topics; subword modeling and spoken term discovery. Unsupervised subword modeling aims to learn speech representations that capture linguistic structures and that are robust for speech recognition. On the other hand, the aim of unsupervised term discovery (UTD) is to find repeating patterns (phonological, lexical or phrasal units) given the raw acoustic signal, without any supervision.

In general, UTD systems take feature time series as input and the output is the discovered clusters of segments, where each cluster is hypothesized to be a unit in that language. For spoken languages, the repeating units may correspond to phones, words or common phrases in that language. Usually UTD systems employ three stages. The first one is the matching stage, in which pairs of similar segments are discovered. The second stage involves the clustering of these pairs, so that similar pairs are joined together to form clusters of hypothesized units. The last stage concerns the parsing of the input sequences with discovered word-type IDs. The performance of the clustering and parsing stages depends on the quality of the matching stage. Hence, it is mandatory to obtain good quality matches before moving on to clustering stage.

In this work, we define a new task for processing of sign language videos. Unsupervised discovery of sign terms is the task of discovering and segmenting sign glosses automatically, without using any supervisory information (additional modalities, lexical knowledge etc.). This work is the first to explore the applicability of unsupervised term discovery algorithms in sign language. This task may provide numerous benefits to sign language and action recognition fields. It can be used as a segmentation tool that proposes gloss time boundaries and it can speed up manual annotation. Moreover, clustered segments can be treated as weak labels and supervised models can be trained based on these labels.

The contributions of this thesis can be summarized as follows:

- The main contribution of this work is to demonstrate that unsupervised term discovery algorithms that are used in speech processing can be used to discover sign terms, if given good quality visual features. It is shown that the SDTW based algorithm [4], a more efficient SDTW (E-SDTW) discovery algorithm [5], and a KNN based algorithm [6] can be run using hand shape features to discover sign glosses in a zero resource setting. The experiments are carried out on a large scale continuous sign dataset and results are evaluated using a set of metrics adapted from spoken UTD task. The experiments with E-SDTW method resulted in a workshop proceeding [7].
- The KNN based UTD algorithm [6] can be successful using two different types of visual features. Thus it is shown that UTD algorithms may be generalized to work with other types of features as well.
- The fusion of the results of different UTD algorithms can improve the discovery performance.
- A cross-validated evaluation scheme for the Phoenix Weather sign dataset is proposed, which might later be used to benchmark different UTD algorithms or feature types.

The rest of this thesis is organized as follows. The related works are presented in Chapter 2. The details of the employed term discovery algorithms are explained in Chapter 3. The setup with regard to the dataset, features and evaluation metrics for sign language experiments are given in Chapter 4. Results of the sign term discovery experiments are discussed in Chapter 5. The conclusions and future directions are discussed in Chapter 6.

2. BACKGROUND

Section 2.1 is dedicated to explanation of Hidden Markov Model training strategies for ASLR. These explanations provide a background for better understanding the works by Koller et al. [1,8,9] which constitute integral parts of this thesis, namely the dataset and feature extraction. It is best to review these works in relation to each other, since they use the same dataset and have commonalities in learning methods. In Section 2.2, unsupervised learning methods that have been applied for sign languages are reviewed. In Section 2.3, the concepts of zero resource speech processing are introduced. In Section 2.4, technical overview of DTW methods which constitute the backbone of employed algorithms in this work are provided for reference.

2.1. Continuous Sign Language Recognition

The recognition scheme for continuous sign language is very similar to ASR. Given a sequence of processed images (or features) x_1^T, the aim is to find the most probable sequence of target terms w_1^N. In general, this problem is formulated by the *maximum a-posteriori* (MAP) decision rule,

$$x_1^T \rightarrow \left[w_1^N\right]_{opt} = \underset{w_1^N}{\operatorname{argmax}} \ \Pr\left(w_1^N|\ x_1^T\right) \tag{2.1}$$

which states that the input sequence is assigned to the term sequence that maximizes the class posterior probability. Using Bayes' theorem, the class posterior can be written as the product of class prior and conditional likelihood

$$\Pr\left(w_1^N|\ x_1^T\right) = \Pr(w_1^N)\Pr(x_1^T|w_1^N). \tag{2.2}$$

The prior is often approximated by n-gram language models, whereas the conditional likelihood term is modelled by the employed generative model. The generative model parameters are learnt during training, by providing the true word sequence w_1^N.

2.1.1. HMM-GMM Training

In traditional setups, the generative visual process have been modelled by Gaussian mixture models (GMM) and Hidden Markov Models (HMM). With the incorporation of hidden states by HMM, the generative visual process $p(x_1^T|w_1^N)$ can be modelled as

$$p(x_1^T|w_1^N) = \sum_{s_1^T} p(x_1^T, s_1^T|w_1^N) \tag{2.3}$$

$$= \sum_{s_1^T} \prod_{t=1}^{T} p(x_t, s_t|x_1^{t-1}, s_1^{t-1}, w_1^N) \tag{2.4}$$

$$= \sum_{s_1^T} \prod_{t=1}^{T} p(x_t|s_t, w_1^N)\ p(s_t|s_{t-1}, w_1^N) \tag{2.5}$$

where the sum in Eq. 2.3 is taken over all possible paths that are allowed by w_1^N word sequence, whereas Eq. 2.4 follows from the chain rule. Since s is not observable, assuming first order Markov dependency between hidden states allows us to write Eq. 2.5 as a product of *observation* model and *transition* model. During the expectation phase of the expectation maximization (EM) algorithm, the most likely path s_1^T is selected using Viterbi training. With the substitution of HMM equations, overall MAP rule becomes,

$$\left[w_1^N\right]_{opt} = \underset{w_1^N}{\operatorname{argmax}} \left\{ p\left(w_1^N\right) \max_{s_1^T} \left\{ \prod_{t=1}^{T} p\left(x_t|s_t, w_1^N\right)\ p\left(s_t|s_{t-1}, w_1^N\right) \right\} \right\} \tag{2.6}$$

in which the observation model $p(x_t|s_t, w_1^N)$ is traditionally modelled as Gaussian mixture. If we let $k := s_t, w_1^N$ represent the state s of word w it can be written as

$$p(x|k) = \sum_{m=1}^{M} c_{m,k}\ \mathcal{N}(x; \mu_{m,k}, \Sigma) \tag{2.7}$$

which means each unique state s_t has its own mixture of multivariate Normal distributions $\mathcal{N}(x; \mu, \Sigma)$ and the mixture weights sum up to 1 such that,

$$\sum_{m=1}^{M} c_{m,k} = 1. \tag{2.8}$$

Typically a global covariance Σ is used in order to ease training. Employing the expectation maximization (EM) algorithm helps us to estimate the GMM parameters c, μ, Σ iteratively. State transition probabilities $p\left(s_t|s_{t-1}, w_1^N\right)$ are generally pooled across words and assigned fixed values. So after training iterations, the frame-state alignments can be used to associate each frame with the corresponding word w, thus achieving forced alignment of true labels.

2.1.2. Hybrid HMM-CNN Training

In hybrid HMM-CNN training, the CNN discriminator $p(k|x)$ is embedded into the observation model $p(x|k)$, by using the Bayes' rule

$$p(x_t|k) = p(x_t)\ \frac{p(k|x_t)}{p(k)} \tag{2.9}$$

where $p(k)$ can be estimated using the state occupation frequencies, assigned during expectation phase. Neglecting the constant evidence $p(x_t)$ we achieve the relation

$$p(x_t|k) \propto \frac{p(k|x_t)}{p(k)} \tag{2.10}$$

which can be inserted into the overall equation together with the scaling hyper-parameter α as

$$\left[w_1^N\right]_{opt} = \underset{w_1^N}{\operatorname{argmax}} \left\{ p\left(w_1^N\right) \max_{s_1^T} \left\{ \prod_{t=1}^{T} \frac{p(k|x_t)}{p(k)^\alpha}\ p\left(s_t|s_{t-1}, w_1^N\right) \right\} \right\} \tag{2.11}$$

thus we can embed the class posterior probabilities obtained by a very complex visual model (CNN) into the generative process. The labels k for training the CNN is obtained by HMM training, where initialization strategies may differ according to application.

2.1.3. Tandem Approach

The tandem approach unifies the HMM-GMM model and CNN in separate steps. However the CNN is not trained in conjunction with HMM, rather a pre-trained CNN is used as a feature extractor. Then these features are used to train the HMM-GMM model. It is computationally more expensive since an extra GMM training is performed in addition to CNN training, but there are cases where this tandem approach proved useful, which we shall in the following sections.

2.2. Unsupervised Methods for Sign Language Processing

Unsupervised learning has been a rather inactive area in sign language recognition. Previous works that focus on lexicon discovery usually rely on weak supervision, in the form of subtitles [10, 11] or text translations [12] that accompany sign videos. A similar work to ours [13] finds common signs among continuous sentences. They make use of the relational distributions to find the co-occurring signs from continuous sign sequences given the information that the sequences share a common sign. An improved version is presented in [14], where they report the system performance based on localization of most common signs in 155 sentences from American Sign Language. Even though they do not use labels, their work differs from ours since they use the knowledge of how many sign segments should be discovered from each sequence beforehand. These works rely on weak supervision or incorporate linguistic information to the discovery process.

From the perspective of sub-unit representation, [15] introduce a sign language phonetic modelling framework in which signs are segmented into dynamic and static sub-units, in an unsupervised fashion. Evaluation of subunit modelling is carried out

with regard to ASLR performance on isolated sign datasets. The works that focus on extracting sub-units [15–17] do not perform discovery at sign level. They are data driven approaches for finding basic sub-units. Doga et al. [18] applies the correspondence autoencoders (CAE) training method to perform unsupervised clustering of key hand shapes from isolated sign sequences. These works focus on finding the smallest sign units, such as hand shapes. However, our aim is to discover longer units such as sign glosses or sequences of signs, from a large scale dataset, without using any supervision.

2.3. Zero Resource Speech Processing

The fully unsupervised methods for learning acoustic models and lexicon discovery are referred as zero resource speech processing. In the recent years, Zero Resource speech challenges [2, 3] were held to allow comparison of various zero-resource approaches using standardized metrics. The two main tasks of these challenges are summarized in the following subsections. Our focus is on the second task, namely unsupervised term discovery. The most notable UTD algorithms are mentioned in Section 2.3.2 and the algorithms that we utilize are explained in great detail in the next chapter.

2.3.1. Unsupervised Subword Modeling

The first task defined in Zero Resource Speech challenges [2,3] is a representation learning task, where the aim is to map the features to a new feature space. The new features should better capture linguistic information and be robust to speaker variance. The quality of the frame-level features are evaluated using the ABX discriminating task. Given two different tokens of phonemic minimal pairs (e.g. A:'beg' - B:'bag') that differ by one syllable and a query token (X:'beg'), the good quality features would result in less DTW distortion between A-X than B-X. In other words, ABX score measures the probability that A-X are closer than B-X, in the representation space. Using this principle, the results are averaged over all A,B,X triplets.

The bottom up approaches cluster the frame level representations to discover acoustic units. The representation learning methods using neural networks are discussed in [19]. The correspondence autoencoders (CAE) [20] use the similar pairs discovered from UTD systems as the input and output for autoencoder networks. The bottleneck representations then serve as features that are robust to non-linguistic variances. Siamese networks [21–23] can be used to learn better embeddings or distances, again using the pairs discovered by UTD systems. Unsupervised learning of fixed-length representation methods are compared in [24, 25]. The CAE idea is extended to sequence to sequence autoencoders with fixed length bottleneck features in [26]. Nonetheless, the focus of this work is the second task, namely unsupervised (spoken) term discovery, therefore we shall discuss notable UTD works in detail in the following sections.

2.3.2. Unsupervised Spoken Term Discovery

The unsupervised spoken term discovery algorithms can be grouped into two categories. The first group of algorithms finds isolated pairs of segments and performs clustering of those segments, thus resulting in partial coverage of the input data whereas the second group of algorithms perform full coverage segmentation of the input sequences, so that each segment is assigned to a cluster label. Notable works that belong to both of these categories are discussed briefly below. Moreover, details of the three partial coverage type UTD algorithms that are used in our work are presented in the following chapter.

2.3.2.1. Partial Coverage Systems. The pioneering work [27] in spoken term discovery introduces the segmental variant of dynamic time warping (SDTW) algorithm to search for pairs of similar segments. The input files are processed in pairs and pairwise distance matrices between their time series feature vectors are computed. The idea is to apply DTW in diagonal bands on a distance matrix between two sequences and collect the path fragments with minimal distortions, so that similar sub-sequences can be paired. In a later version [4], the discovered diagonal path fragments with

high similarities are clustered to form hypothesized word categories. A similar but more efficient algorithm [5] uses locality sensitive hashing to approximate distance matrices. The diagonal fragments with high similarities are searched using efficient image processing techniques. Costly SDTW search is applied only in the vicinity of these candidate fragments, thus reducing runtime significantly.

The system proposed by Muscariello et al. [28] builds a library of templates as the input stream is processed. The candidate query segments are compared to a library of motifs that are previously discovered. If there are no similar motifs, the segment is compared to the search buffer. Thus if a pattern is not in the library, it can be discovered by matching it with another pattern from the past buffer stream. The search buffer is composed of the past stream of a certain duration which is determined by the expected period for a unit to repeat itself.

2.3.2.2. Full Coverage Systems. The systems that perform full coverage segmentation into word labels usually employ Bayesian models to group together similar acoustic units. The pioneering work by Lee & Glass [29] uses unsupervised HMM training to segment and learn phonetic units. The model is a hierarchical Dirichlet process Gaussian Mixture Model (DPGMM) [30] where the HMM components are used to model the sub-word units. The inference is carried out using Gibbs sampling. Heck et al. [31] uses linear discriminant analysis to better map feature space for DPGMM clustering. A similar approach is proposed by Ondel et al. [32], where they employ a variational Bayes procedure [33] for inference. This work is used as the baseline acoustic unit discovery system for ZR 2019 Challenge [34]. The HMM alignments found by DPGMM method can be used to train a HMM-VAE models [35, 36]. The method proposed by Kamper et al. [37] uses fixed-dimensional embeddings to represent variable length sequences instead of frame level representations and HMM's.

The Bayesian methods [32, 37] that perform full-coverage, require large amounts of data to be trained and are usually fit to represent phoneme like sub-word units. The number of phonemes in a spoken language is usually limited around 50 and different

tokens of the same phoneme do not possess much variability compared to tokens of sign language words. Our focus in this work is to discover longer, word-like units, therefore we have not considered full-coverage systems in our experiments.

2.3.3. Evaluation Metrics

The term discovery systems can be made up of three stages, namely matching, clustering and parsing stages. These stages can be evaluated using different metrics, which are described in detail by Ludusan et al. [38]. The metrics are computed using the gold transcriptions of discovered segments. In Zero Speech challenges [2, 3], the transcriptions are usually at phoneme level and a segment is associated with gold phones if the segment interval overlaps with more than 50% or 30ms of the phone duration. The related metrics for each of these stages are summarized below. Moreover, the metrics that we use for sign language are explained in detail in Chapter 4.

2.3.3.1. Matching Stage. The matching stage outputs a set of pairs of similar segments. The evaluation of this stage concerns normalized edit distance (NED) and coverage. NED measures the quality of pairs, in terms of Levenshtein distance, which simply is the minimum number of modifications (insertion, deletion, substitution) required to make two discrete sequences the same, normalized by the length of the longer sequence. The final NED score is averaged over all pairs. Coverage is the ratio of non-overlapping discovered tokens to the discoverable tokens in all input sequences. It is computed as discovered phones over all discoverable phones.

2.3.3.2. Clustering Stage. The clustering stage is also referred as the *lexicon discovery* stage, in which the matched pairs are grouped into clusters. The clusters become the hypothesized lexical types in that language. The evaluation of this stage is done using grouping and type quality metrics, where the metrics are defined in terms of precision, recall and F-score. The grouping quality is similar to cluster purity and inverse purity. If the pairs within a cluster has the same transcription, then the precision is high. If

pairs from separate clusters have the same transcriptions, then the recall is low.

Type quality metrics analyse whether discovered groups correctly represent the lexical types. If the discovered fragments segment the words incorrectly, type scores are going to be poor even though the clusters may be pure.

2.3.3.3. Parsing Stage. This stage is also named as the *word segmentation* stage, in which the input stream is transcribed and segmentation boundaries are assigned. The token and boundary qualities are again measured in terms of precision, recall and F-score. The token quality measures the amount of word tokens that are correctly segmented. The boundary quality is measured such that if a proposed boundary is within close proximity of a gold boundary, it is counted as correct. So, these metrics give an idea of the quality of word token segmentation.

2.4. DTW Overview

Dynamic time warping is a well-known algorithm that is used for achieving non-linear alignment of two time series, which might have different durations. This enables the similarity comparison of signals, while accounting for small variations. Segmental DTW is a variant of the well known dynamic time warping algorithm and it is introduced by Park and Glass [4]. The following sections describe the details regarding the DTW algorithm and the segmental DTW variant.

2.4.1. Dynamic Time Warping

Given a pair of vector time series as $\mathcal{X} := (\mathbf{x}_1, ..., \mathbf{x}_{N_x})$ and $\mathcal{Y} := (\mathbf{y}_1, ..., \mathbf{y}_{N_y})$, distortions computed using a frame-wise cost function d, which may or may not be a distance metric. The alignment path ϕ gives the warping relation between two time series. More formally, a warping relation is a sequence of ordered pairs,

$$\phi = \{(i_k, j_k)\} \quad , \quad k = 1, ..., T \tag{2.12}$$

where the mapped elements are $\mathbf{x}_{i_k}$ and $\mathbf{y}_{j_k}$. In the case of global alignment, ϕ_1 is $(1,1)$ and ϕ_T is (N_x, N_y). Another constraint,

$$\phi_k - \phi_{k-1} \in \{(0,1),(1,0),(1,1)\}$$

ensures that the alignment is monotonic. With these constraints, the optimal alignment path is the one that minimizes the total distortion

$$D_\phi(\mathcal{X},\mathcal{Y}) = \sum_{k=1}^{T} d(\mathbf{x}_{i_k}, \mathbf{y}_{j_k}) \tag{2.13}$$

where distortion matrix is computed using dynamic programming techniques, which keeps track of minimum accumulated distortion, as described in Figure 2.1. Then starting from $D(N_x, N_y)$, optimum path ϕ^* is found by using back-tracking.

DTW(x,y)

$D(1,1) \Leftarrow 0$ $\quad \triangleright$ Initialize the distortion matrix, D

for $j = 2$ to N_y **do**

  $D(1,j) \Leftarrow D(1,j-1) + d(\mathbf{x}_1, \mathbf{y}_j)$

end for

for $i = 2$ to N_x **do**

  $D(i,1) \Leftarrow D(i-1,1) + d(\mathbf{x}_i, \mathbf{y}_1)$

end for

$\triangleright$ Accumulate distortions

for $i = 2$ to N_x, $j = 2$ to N_y **do**

  $\mathcal{T} \Leftarrow \{(i-1,j-1),(i,j-1),(i-1,j)\}$

  $t^* \Leftarrow \text{argmin}_{t\in\mathcal{T}}\,(D(t))$

  $D(i,j) \Leftarrow D(t^*) + d(\mathbf{x}_i, \mathbf{y}_j)$

end for

Figure 2.1. Dynamic Time Warping Algorithm.

2.4.2. Segmental Dynamic Time Warping

The DTW algorithm described above (Figure 2.1) tries to find a global alignment path by considering the totality of sequences. However if the task is to find similar sub-sequences, then global alignment fails. For example, if we compare two sequences of features corresponding to the sentences

(i) The Chinese markets are open this weekend.
(ii) These firms are subsidized by the Chinese government.

with the aim of finding the common word 'Chinese', a global alignment would map the features of different words to each other. This is where the segmental dynamic time warping (SDTW) algorithm proves to be useful. In SDTW, multiple alignment paths are searched in different diagonal bands. This is achieved by imposing locality constraints on the warp paths as well as starting the search from different seed points. The locality constraint, introduced by Sakoe et al. [39], forces the k^{th} coordinate of the warp path to obey

$$|(i_k - i_1) - (j_k - j_1)| \leq R \tag{2.14}$$

for the starting coordinate (i_1, j_1), where R is the width of the diagonal search band. The locality constraint together with these starting coordinates ensure that the warp paths are inside diagonal bands. The set of start coordinates $\{(i_k, j_k)\}$ are chosen to be the union of

$$\left\{ \big((2R+1)k, 1\big) \quad | \quad 0 \leq k \leq \left\lfloor \frac{N_x - 1}{2R+1} \right\rfloor \right\}$$

and

$$\left\{ \big(1, (2R+1)k\big) \quad | \quad 0 \leq k \leq \left\lfloor \frac{N_y - 1}{2R+1} \right\rfloor \right\}.$$

3. METHODS

This chapter is focused on the UTD algorithms that are used in our experiments. These algorithms are originally intended to be applied in spoken language, using speech features. Nevertheless, we make use of these algorithms using sign language features and our term discovery pipeline is almost the same once the features are extracted. Therefore, throughout Section 3.1, the algorithms are described as general purpose term discovery algorithms given the time series feature vectors. Section 3.2 describes the various strategies for grouping the pairs of segments.

3.1. Pairwise Discovery

In this section, the matching stages of the three spoken UTD algorithms [4–6] are described in great detail. The first algorithm that we describe is the pioneering spoken UTD algorithm, referred as SDTW. The second one is a more efficient version (E-SDTW) proposed by Jansen and Van Durme [5]. The last one is a KNN based algorithm [6]. The inputs to the matching stage are the feature vector time series and the output is a collection of segment pairs, with accompanying matching scores.

3.1.1. Segmental DTW Based Discovery

The SDTW algorithm [4] compares two sequence of vectors, which are typically at sentence level in UTD applications. Given a set of sequences for UTD task, segmental-DTW search is run for all possible pairs of sequences. This yields a set of local alignment paths for each pair combination. Then, shorter fragments with low costs are isolated from these paths. The low cost fragments are used to construct an adjacency graph, where nodes correspond to the low cost segments from the signal and edge weights are inversely proportional to the warping cost between the segments. Using this graph, a community detection algorithm is run to discover clusters of similar segments. At the end, the clusters represent the hypothesized word types and the segments of each

cluster represent the word tokens. These steps are explained below in more detail.

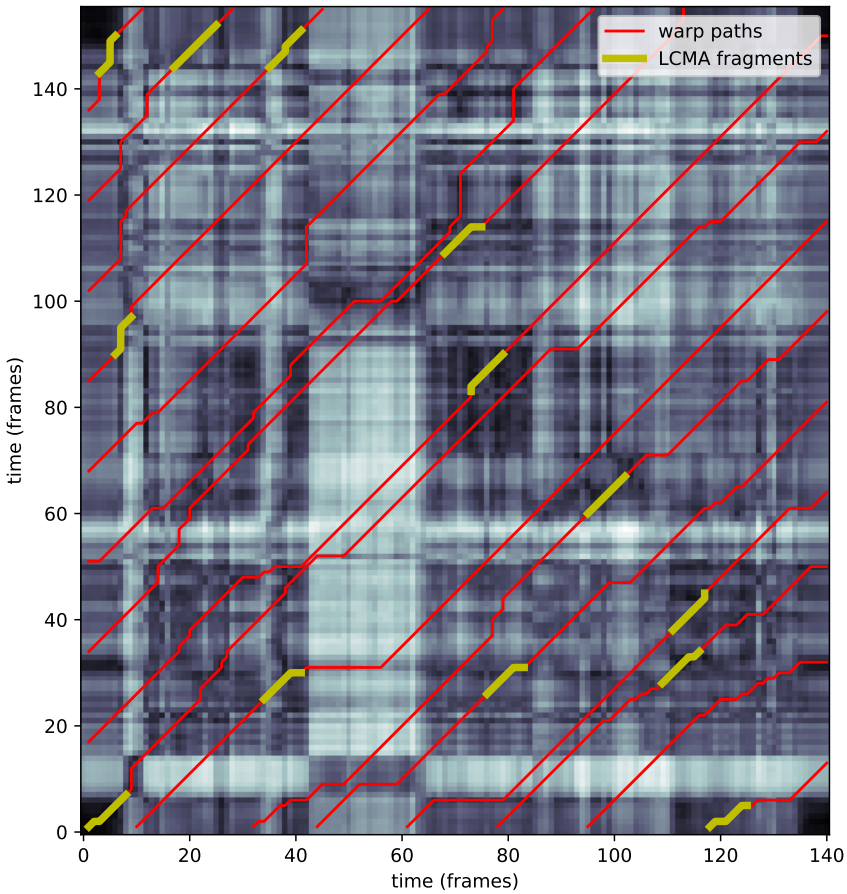


Figure 3.1. Example of SDTW warp paths, searched on a pairwise cost matrix computed from features of two sign language sentences.

Given a set of feature sequences that can have variable lengths, pairs of sequences are selected for segmental-DTW search. If we denote these two vector time series as $X \in \mathbb{R}^{d \times N_x}$ and $Y \in \mathbb{R}^{d \times N_y}$, a pairwise distance matrix $M \in \mathbb{R}^{N_x \times N_y}$, is computed using cosine distance. Then, DTW alignment paths are searched in non-overlapping diagonal bands over M. These searches start from different seeds and are subject to locality constraints. Here, locality threshold is denoted by w, which determines the width of the diagonal band. Once the local alignment paths are obtained, a fragment of

minimum length L_{min} with the lowest average cost is isolated for each path (as shown in Figure 3.1). Repeating this for all pairs of input sequences, we end up with a set of candidate fragments. From these isolated fragments, a fraction of θ with the lowest alignment costs are retained and the rest are discarded. Then, node centers are found by selecting the peaks from similarity profile as described in [4].

After warping paths ϕ_r^* for each diagonal band r are found, the next step is to discard the parts of the paths that have high distortion. This is achieved by finding the length constrained minimum average (LCMA) fragment from each path. Given a sequence of real numbers

$$S = (s_1, s_2, ..., s_N)$$

and a length constraint L_{min}, the length constrained minimum average fragment is the consecutive sub-sequence whose length is at least L_{min} elements and which has the minimum average. The problem can be stated formally as finding the start and end indices, m^*, n^*, of the fragment such that

$$m^*, n^* = \underset{1 \le m \le n \le N}{\operatorname{argmin}} \quad \frac{1}{n-m} \sum_{k=m}^{n} s_k \tag{3.1}$$

which is subject to $n - m \geq L_{min}$. The original work [4] uses the LCMA algorithm of [40], whereas we implemented a slower version for the sake of implementation simplicity. The LCMA algorithm that we use is described in Figure 3.3. Using this algorithm, warp paths are refined to shorter fragments that present the highest similarity. So given an alignment path

$$\phi = \{(i_k, j_k)\}, \quad k = 1, ..., T$$

and the corresponding alignment cost

$$D_\phi = \{d(\mathbf{x}_{i_k}, \mathbf{y}_{j_k})\}, \quad k = 1, ..., T$$

the $LCMA\,(D_\phi, L_{min})$ function yields the onset and offset indices m^*, n^* together with the minimum average path distortion μ_{min}.

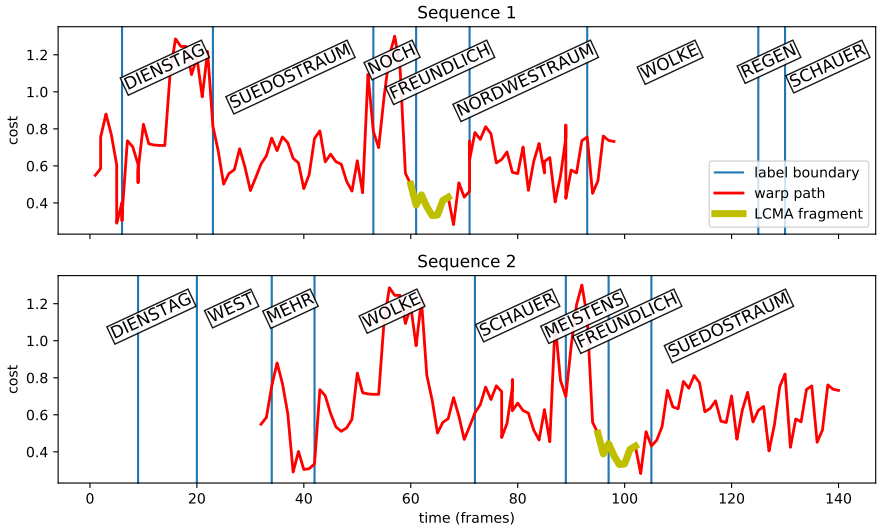


Figure 3.2. Example of SDTW warp paths, obtained by comparing sign feature sequences belonging to two sentences from RWTH Phoenix 2014 dataset.

Example of LCMA fragments are shown as yellow lines on top of red warp paths in Figure 3.1 and Figure 3.2. As it can be seen from Figure 3.2, the LCMA fragment corresponds to the same gloss type, 'FREUNDLICH', along the warp path. The segmental DTW makes the search in different regions so it enables the discovery of subsequences. The value of L_{min} controls the granularity of the discovered units, where smaller L_{min} would likely to result in many subunit-like segments and larger L_{min} would result in fewer word or phrase-like segments.

Running these steps for all input pairs $(\mathbf{x}_i, \mathbf{x}_j)$, the result is a collection sub-sequence pairs $\mathcal{P} = \{(s_p^1, s_p^2, \mu_p)\}_{p=1}^{P}$, where each sub-sequence s_p^i is identified by a triple (f_p^i, m_p^i, n_p^i) that contains the unique input file identifier f_p^i for i^{th} file, onset and offset time indices (m_p^i, n_p^i), whereas μ_p is the associated warping cost. These pairs of segments are referred as *matching segment pairs* throughout the rest of this work.

LCMA (sequence S, constraint L_{min})

$C \Leftarrow S(1)$

for $n = 2$ to N **do**

  $C(n) \Leftarrow C(n-1) + S(n)$ ▷ *Compute cumulative sum vector* C

end for

$m^* \Leftarrow 1, n^* \Leftarrow N$

$\mu_{min} \Leftarrow (C(N) - C(1))/N$

for $m = 1$ to N **do**

  for $n = m + L$ to N **do**

    $\mu_{tmp} \Leftarrow (C(n^*) - C(m^*))/(n - m)$

    if $\mu_{tmp} < \mu_{min}$ **then**

      $\mu_{min} \Leftarrow \mu_{tmp}$

      $m^* \Leftarrow m; n^* \Leftarrow n$

    end if

  end for

end for

return m^*, n^*, μ_{min}

Figure 3.3. Length Constrained Minimum Average Algorithm.

3.1.2. Efficient SDTW Baseline Algorithm

Despite being fairly successful, the SDTW based discovery algorithm of Park and Glass [4] had one drawback: it was not scalable because of the $O(n^2)$ time complexity. In order to combat this shortcoming, Jansen et al. [5] proposed a more efficient algorithm, which we shall refer as *Efficient-SDTW* algorithm throughout this thesis. Efficient SDTW algorithm is said to compromise accuracy for speed by achieving $O(n\log n)$ time complexity. There are important modifications to the discovery step, which involve approximations at different stages. The details for these steps are explained below.

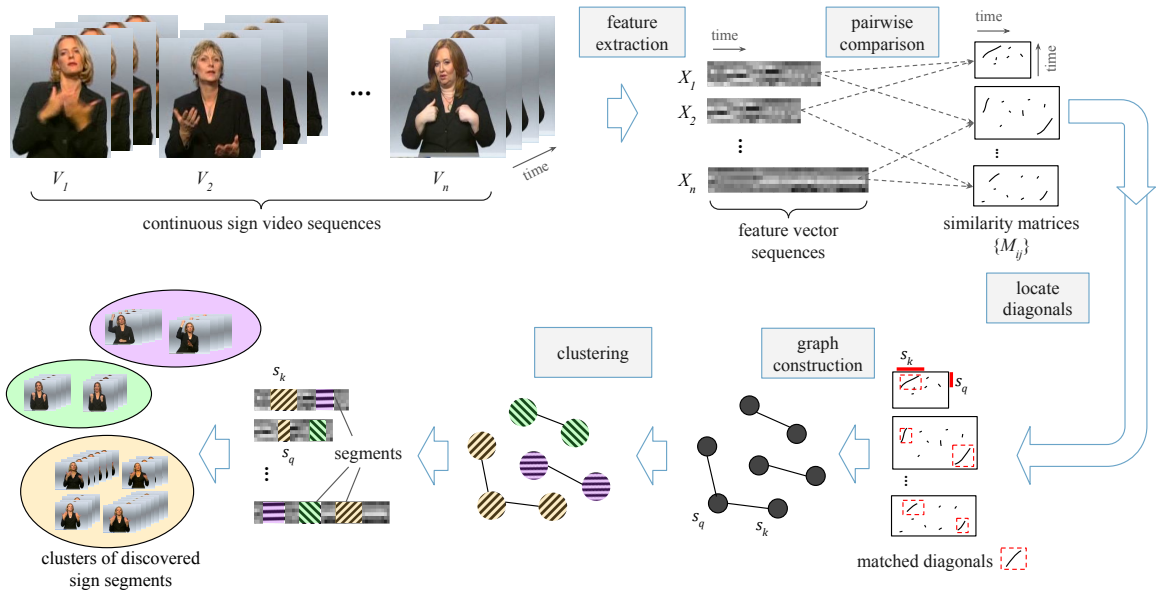


Figure 3.4. Flow diagram of the efficient SDTW based discovery algorithm.

The discovery step of the E-SDTW algorithm [5] expects the same kind of input as the first algorithm [4]; a set of feature vector time series for each utterance, $\mathcal{X} := \{\mathbf{X}_i \mid i \in \{1, ..., N\},\ \mathbf{X}_i \in \mathbb{R}^{d \times T_i}\}$, where d is the number of feature dimensions and T_i is the length of each sequence and N is the total number of input utterance. Again, the input sequences are compared in pairs. However, the novelty begins with the computation of the similarity matrix $M \in \mathbb{R}^{T_i \times T_j}$. Instead of filling all of the elements by making $T_i \times T_j$ comparisons, the matrix M is populated sparsely, by taking into account only the comparisons that are likely to yield high similarity. This is achieved

by making use of randomized algorithms: namely locality sensitive hashing (LSH) and point location in equal balls (PLEB), which are to be explained in the following sections. The second novelty is to treat M as an image and search the similar sub-segments by efficient image processing techniques, avoiding the exhaustive SDTW search as done in [4]. Only after the candidate matches are found, the SDTW search is performed locally around these regions, thus limiting the expensive $O(n^2)$ search area.

3.1.2.1. Similarity Matrix Approximation. The main idea behind the approximation of the similarity matrix relies on using the Hamming distances of the hashed features (LSH), as an approximation for the cosine similarity and then, using the sorted hash signatures to compute pairwise similarity only between close vectors (PLEB). The feature vectors $\mathbf{x} \in \mathbb{R}^d$ are hashed to b dimensional *bit signatures* $h(\mathbf{x}) \in \{0,1\}^b$ using random projections. A random transformation matrix $\mathcal{T} \in \mathbb{R}^{d \times b}$, whose elements are drawn from $\mathcal{N}(0,1)$, is used to project $\mathbf{x}$ onto $\mathbb{R}^b$. Then, the elements of the projected vector are mapped to 0's and 1's according to their signs and the result is the hashed features $h(\mathbf{x})$. This transformation enables the approximation of the cosine similarity by the following relation,

$$\cos(\mathbf{x}_i, \mathbf{x}_j) \approx \cos\left(\frac{H\left(h(\mathbf{x}_i), h(\mathbf{x}_j)\right)}{b}\pi\right) \tag{3.2}$$

where $H(h_x, h_y)$ is the Hamming distance between two bit signatures h_x, h_y and simply gives the number of different bits. The relation achieves equality as b goes to infinity.

The bit signatures are used in PLEB algorithm which performs a fast approximate nearest neighbour search. The algorithm may be run for P iterations. In each iteration, the bit orders of signatures are shuffled, and the set of shuffled signatures are sorted. The purpose of the sorting comes from the idea that if two signatures are similar to each other, their sorted signatures will fall within a close proximity. Therefore each signature in the sorted lists is compared to B of its nearest neighbours. If the similarity exceeds a threshold δ, the corresponding entry of the M is populated with the similarity value. Repeating the shuffling and sorting steps for P iterations decreases the approximation

error. An additional diagonal search with probe length D is performed around the high similarity points, in order to compare $\pm D/2$ temporal neighbours.

The time complexity of this routine for an input of length n can be derived as follows. For a single iteration, the sorting of n signatures costs $O(n\log n)$ time, whereas the beam search costs $O(nB)$. Repeating this P times makes the overall complexity $O(P(n\log n + nB))$ time. The complexity approaches $O(n\log n)$ as n get much larger than B and P, whose values are typically in the order of 10.

3.1.2.2. Finding the Matching Fragments Using Image Processing. With the PLEB routine, the elements of M are populated sparsely. This routine introduces some noise to the similarity matrix. The idea is to treat the M as an image and locate diagonal segments using well-known image processing techniques.

The first step in this routine is to binarize M according to the similarity threshold δ, to obtain the binary matrix M'. Then, a diagonal 1-dimensional μ-percentile filter of length d_x, with orientation *parallel* to the target lines, is applied to reduce the noise in diagonal lines. A μ-percentile filter is a more general form of the median filter, where μ can take values between 0% and 100%, whereas $\mu = 50\%$ for the median filter. Next, a 1-dimensional Gaussian smoothing filter of length d_y is applied in the orientation *orthogonal* to the target lines, in order to allow variations in articulation speed of the same word. After these filtering operations that make the diagonal line segments more apparent, the next step is to locate these diagonal segments using Hough transform. A Hough transform, whose orientation angle is fixed at $-45°$ with respect to the x-axis, is used to accumulate the number of pixels that are projected onto $y = -x$ line. The peaks of the accumulations that are greater than ρ, gives the locations of the diagonal rays.

3.1.2.3. Path Refinement with Local SDTW. The previously described routines make up the first pass of this algorithm [5]. The second pass proceeds with the time consum-

ing SDTW search ($O(n^2)$) on the located diagonal segments. For each line segment, SDTW search is carried out both forward and backward in time, where both searches start from the mid-point of the located diagonal. The searches are limited to a diagonal band of width R frames. For both directions the search is terminated when the path integral of $1 - M_{ij}$ exceeds a distortion budget $C/2$. The last operation trims the points on the two ends of the path, if their similarity is less than T_{trim}. At the end, the discovery step outputs a set of matching pairs of segments, $(s_p^1, s_p^2, \sigma_p) \in \mathcal{P}$, where $\sigma_p \in [0, 1]$ is the similarity score of each match.

3.1.3. K-Nearest Neighbours Based Discovery

We adopt the KNN based discovery pipeline in [6], which begins with extracting large number of overlapping segments from the input sequences. These segments, which may have variable lengths, are transformed into fixed dimensional representations using smoothed sampling. Then, for each segment representation, k nearest segments are searched so that, each segment is paired with k other segments. From these segment pairs, the ones that overlap and that have lower similarities are discarded. The remaining pairs are the discovered pairs. The flow diagram of this algorithm is displayed in Figure 3.5 and details of these steps are explained in the following sections.

3.1.3.1. Temporal Segmentation. For an input sequence, the points that are a frames apart are selected as candidate segmentation points. The segments are extracted for all possible combinations of these candidate points. As the parameter a decreases, the chance of finding correct boundaries increases at the expense of more computational cost. The segment lengths are constrained to an interval, which can be adjusted according to the expected term lengths.

More formally, for a given d dimensional feature vector time series $X \in \mathbb{R}^{d \times T}$ of length T, a set of segments $\{X_{ij}\}$ are extracted such that

$$m, n \in \{0,\ a,\ 2a,\ \ldots\ , \lfloor T/a \rfloor \cdot a\} \tag{3.3}$$

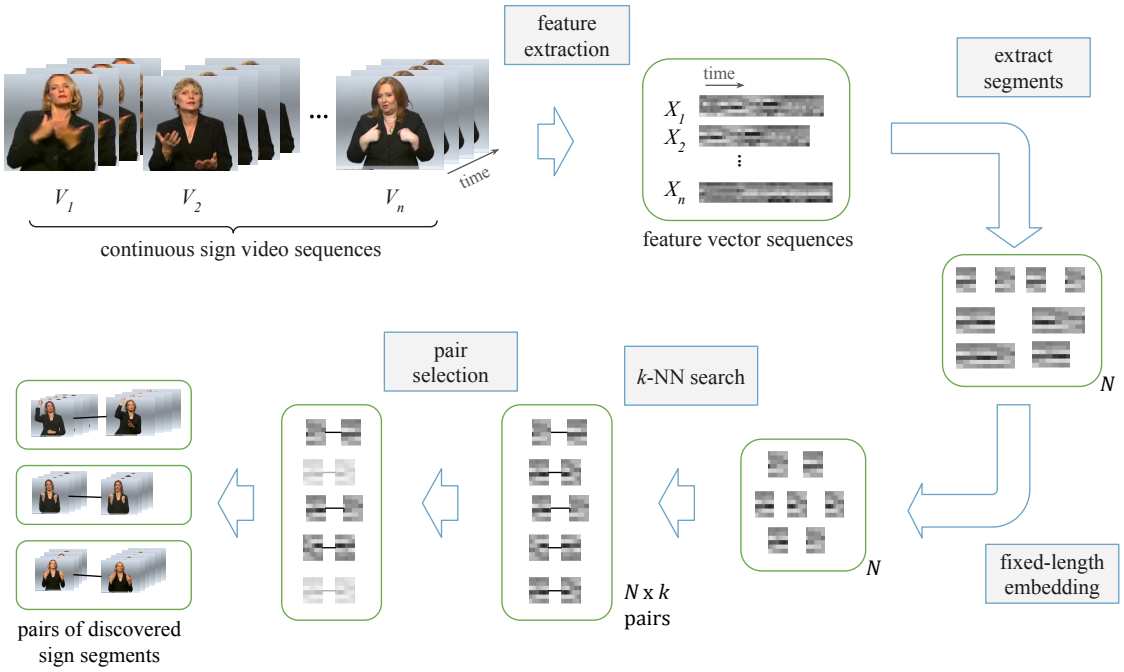


Figure 3.5. Flow diagram of the KNN based discovery algorithm.

$$l_{min} < n - m < l_{max} \tag{3.4}$$

where l_{min} and l_{max} set the bounds for segment lengths. This procedure considers all possible segments as candidates.

3.1.3.2. Fixed-Length Representations. We apply the embedding method described in [6], which simply is the sampling of input vectors, weighted by Gaussian kernels. A segment X_{ij} of L_0 frames is multiplied with a transformation matrix $F \in \mathbb{R}^{L_0 \times L}$ to be mapped to L-frame representation. The l^{th} column of F is the kernel defined as

$$f_l = \mathcal{N}\left(\frac{l \cdot L_0}{L}\ ,\ r \cdot L_0 + s \cdot g_L(l)\right) \tag{3.5}$$

where $g_L(l)$ is a triangular function such that $g_L(l) = \frac{L}{2} - \left|\frac{L}{2} - l\right|$, r and s are weighting parameters for the kernel's variance. The triangular function makes the frames in the middle smoother. This is a very simple method for obtaining fixed dimensional

representation and more complex representation learning methods can be incorporated to this step.

3.1.3.3. Nearest Neighbour Search. Fixed-length representations are reshaped to 1d vectors so that each segment is represented by one of these vectors. The next step is to collect all of them to a search index, using the FAISS [41] framework, which builds a very efficient search index on GPU and can be scaled up easily. Then, for each segment representation, the k nearest segments are found using Euclidean distance. If there are N segments, the search yields $N \times k$ pairs of similar segments.

3.1.3.4. Pair Selection. The pairs after the KNN search are mostly redundant because they overlap with each other. Therefore a series of elimination steps are required, so that only non-overlapping high confidence matches remain. The first step is to retrieve and sort all neighbours for an input file, and select only the top ζ percent of the pairs. In other words, for an input file i, there are $N_i \times k$ pairs and we select the best $\zeta \times N_i \times k$ pairs. The next step is to remove the self-overlapping pairs, whose segments overlap with each other. For a pair $p = (s_1, s_2)$, the self-overlap ratio between the segments s_1, s_2 is computed as the lengths of intersection over union

$$r_{self}(s_1, s_2) = \frac{|s_1 \cap s_2|}{|s_1 \cup s_2|}. \tag{3.6}$$

The final step removes the remaining overlapping pairs by using non-maximal suppression (NMS). All pairs are sorted by decreasing similarity scores. Beginning from the top, the pairs are compared to worse pairs. Worse pairs are discarded if they overlap with a better pair, where the pair-overlap ratio is computed as

$$r_{pair}(p_i, p_j) = \frac{|s_{i,1} \cap s_{j,1}| \cdot |s_{i,2} \cap s_{j,2}|}{|s_{i,1}| \cdot |s_{i,2}|}. \tag{3.7}$$

The ζ parameter is adjusted to control accuracy-coverage trade-off in the original work [6]. In our implementation, we fix $\zeta = 5\%$ and apply another similarity threshold θ at the very end, in order to perform NMS only once and adjust the coverage after all computations are complete.

3.2. Clustering

All of the algorithms described in Section 3.1 have a matching stage, where pairs of similar segments are discovered. Then follows the clustering stage, where the pairs are clustered into larger sets which are hypothesized to be word-like units. Although the clustering methods employed by SDTW [4] algorithm and E-SDTW [5] algorithm are different, they can be applied interchangeably to any set of matching pairs. Therefore, the clustering methods are described with more detail in Section 3.2, as a separate section of their own.

3.2.1. Pairs as Clusters

As discussed earlier, the discovery step yields pairs of matching segments $(s_p^1, s_p^2, \sigma_p) \in \mathcal{P}$. Then, these segments are clustered to obtain hypothesized unit types. If the discovery step can not find good quality matches, then no clustering algorithm can find pure clusters. Therefore it is best to evaluate these stages separately and shift our focus to improving the discovery stage.

In order to evaluate the performance of discovery algorithms, we need to skip clustering step and treat each matching pair as a cluster on its own, as suggested in [38]. Then, there will be as many cluster centers as there are matches. However the pairs of matching segments $(s_p^1, s_p^2, \sigma_p) \in \mathcal{P}$ may overlap. If we do not eliminate the overlapping segments, very similar matches may dominate the overall performance. For this reason, we apply the pair selection strategy described in Section 3.1.3.4 to *de-duplicate* the matches.

3.2.2. Clustering with Word Centers

This clustering strategy is originally intended to work with the pairs obtained from SDTW based discovery [4]. However, we apply it on the matches of the other two algorithms [5, 6] as well. This clustering method will be referred as *Segment Centers* (SC) throughout the rest of this work.

The first step is to use the matching fragments and their associated matching costs to build an adjacency graph $G = (V, E)$, where V is the set of nodes and E is the set of weighted edges. The edge weights are inversely proportional to the warping cost between the nodes.

3.2.2.1. Adjacency Graph Construction. The first step is to convert the average alignment costs μ_p of the sub-sequence pairs to similarity scores σ_p. This is accomplished using the following formula,

$$\sigma_p = \begin{cases} \dfrac{\theta - \mu_p}{\theta} & , \quad \text{if } \mu_p \leq \theta \\ 0 & , \quad \text{otherwise} \end{cases} \tag{3.8}$$

which applies an inverse transformation to retain a certain ratio r_θ of the matches that have average distortion less than θ. This enables us to continue with only the high confidence matches. As shown in Figure 3.6, the idea is to represent all of the matching pairs in a graph structure, where segments s_p^i correspond to vertices (nodes) and similarity scores σ_p correspond to edge weights in this representation.

Two segments are treated as the same node, if they overlap significantly. The overlaps are determined via what Glass et al. [4] call *similarity profile* for each input sequence. A similarity profile for an input sequence gives the time instances that are most likely to be part of a repeating unit. This is computed by aggregating the similarity scores associated with each time instance that may be part of matching pairs. More formally, for an input sequence $\mathbf{x}_i = \{x_1, ..., x_T\}$ the aggregated similarity score

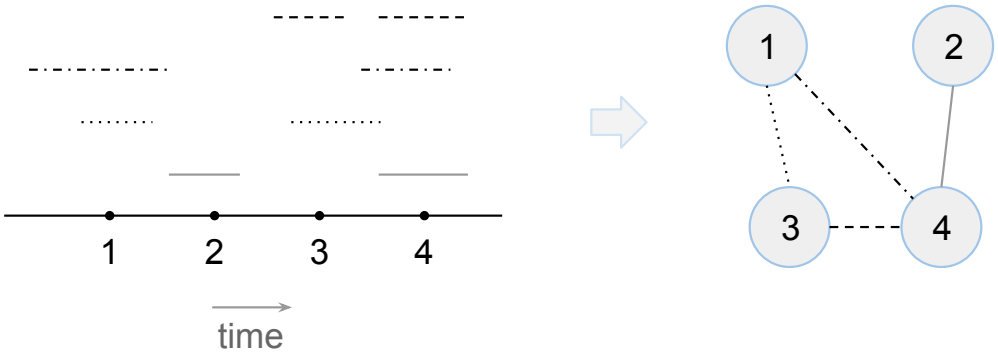


Figure 3.6. An adjacency graph is formed using the pairs of matching segments.

at time t_i is given by

$$S(t_i) = \sum_{(m,n)\in s_p,\, s_p\in\mathcal{P}} \mathbb{1}_{\{m,\ldots,n\}}(t_i)\ \sigma_p \tag{3.9}$$

where $\mathbb{1}_{\{m,\ldots,n\}}(t_i)$ is the indicator function that is 1 when the time index is included in the interval (m, n) for that match and 0 otherwise. Then, it can be deduced that the regions where the similarity profile is high are the regions that are more likely to contain a common word. For example in Figure 3.7, similarity profile for a sign sequence is shown in blue lines, where the y-axis gives the aggregated similarity of each time index. The intuition is, if some regions in the input sequences are contained in multiple matches, these regions are likely to include a repeating unit.

These similarity profiles are smoothed with a triangular window of length 0.5 seconds as shown in Figure 3.7, so that the peaks become more obvious. These peaks are the estimated word centers and used as *nodes* $v \in V$ in the adjacency graph G. Edge weights $e_{ij} \in E$ are assigned between two nodes (v_i, v_j), if there are matching pairs $(s_p^1, s_p^2, \mu_p) \in \mathcal{P}$ whose time intervals $(m_p^1, n_p^1), (m_p^2, n_p^2)$ include v_i and v_j respectively. Edge weights are aggregation of similarity scores σ_p of the related match pairs, if more than one pair overlap both nodes. Finally, an adjacency graph is obtained where the

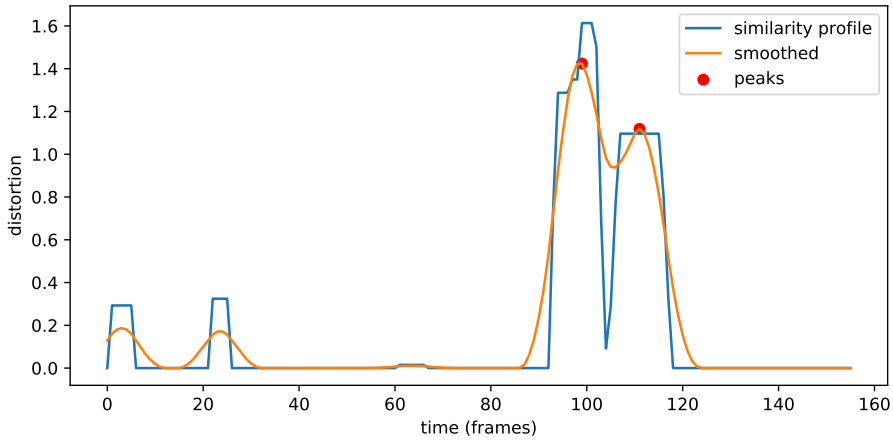


Figure 3.7. Similarity profile for a sign sequence. The peaks represent estimated word centers.

nodes are possible word centers and the edges are the measure of their similarity.

3.2.2.2. Graph Clustering. The next step is to group similar nodes together. Park and Glass [4] suggest using a modularity based community detection algorithm that is developed by Newman [42]. An adjacency graph is constructed in which the edge weights are inversely proportional to warping cost between the nodes. Using this graph, a community detection algorithm is run to discover clusters until a modularity condition q is satisfied. Modularity is a measure of the strength of division of a graph into clusters. High modularity indicates intra-cluster edges are fewer than inter-cluster edges. Once the clusters are found, time intervals for each node center are calculated by averaging the start and end times of the alignment paths which are in the same cluster and include that node center. At the end, the UTD algorithm outputs the clusters of segments (nodes), where each segment is specified by the related filename and time interval.

3.2.2.3. Finding Segment Boundaries. After the clustering step, each vertex v_i is assigned to a cluster C. At this stage, the vertices correspond to single time instances, that are supposed to be centers of words. In order to obtain the word boundaries, the time intervals of the original segments that include the clustered word centers are retraced. The average of onset and offset indices are calculated to determine the boundary of a word corresponding to a graph node.

3.2.3. Clustering by Joining Overlapping Segments

The initial work of Jansen and Van Durme [5] does not include the clustering step. The details of the clustering step are explained in another paper [43], which we shall discuss briefly in the following paragraphs. This clustering strategy will be referred as *Overlap Edge* (OE) throughout the rest of this work. An illustration of the steps for this strategy is given in Figure 3.8.

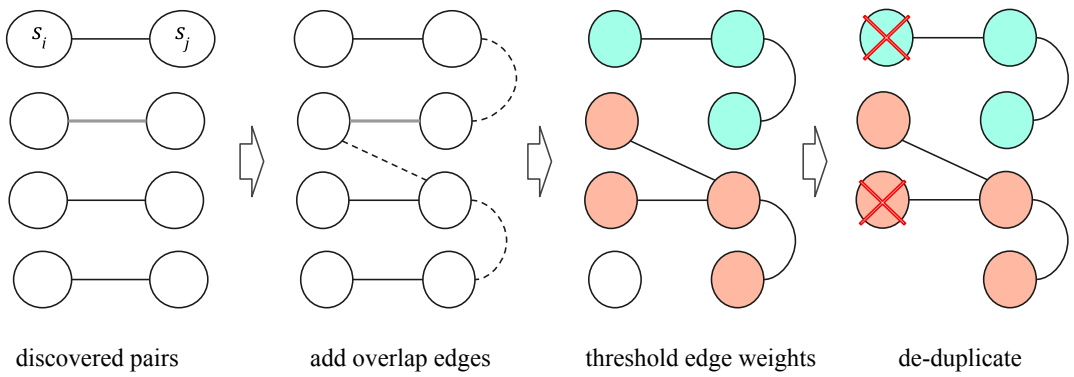


Figure 3.8. Illustration of the steps of overlap edges and connected components clustering method.

The matching pairs are thresholded with T_σ according to their similarity scores before the clustering step, in order to eliminate poor matches and proceed with the confident ones. An adjacency graph is constructed from the confident matching pairs, by adding two kinds of edges between the vertices that represent the discovered segments. The *acoustic* edges are simply the edges that connect the matching pairs

$(s_p^1, s_p^2, \sigma_p) \in \mathcal{P}$, where the edge weights are proportional to the similarity scores σ_p. Furthermore, the *overlap* edges are added between the vertices, if the corresponding segments s_k^i, s_l^j are in the same file and overlap more than a certain T_{olap} threshold. The overlap fraction is given as

$$F_{olap}(s_k^i, s_l^j) = \max(0, r(s_k^i, s_l^j) - 1) \tag{3.10}$$

for

$$r(s_k^i, s_l^j) = \frac{n_k^i - m_k^i + n_l^j - m_l^j}{\max(n_k^i, n_l^j) - \min(m_k^i, m_l^j)} \tag{3.11}$$

where (m, n) are onset and offset indices of the related segments.

Once the edges are assigned between vertices, clustering using simple connected components is carried out. The edges whose weights are less than T_{cc} are removed from the graph and vertices are clustered together if they are connected to each other. At the end the resulting clusters may contain overlapping segments. Therefore the clusters are *de-duplicated* as a last step, such that the segments that overlap more than T_{dedup} with another segment from the same cluster are removed.

4. DATA AND EXPERIMENTAL SETUP

We test the term discovery algorithms described in the previous chapter on sign language videos, by feeding visual features instead of acoustic features. The visual features consist of hand shape and pose features which are obtained by running pre-trained models on each video frame. We use the Phoenix Weather 2014 [1] dataset, in which the gloss time boundaries are labelled, enabling us to evaluate the quality of discovered segments. Using the metrics for spoken UTD [38], we compare the performances of three UTD algorithms and comment on their differences. In the following sections, the dataset, features and experiment setups are explained. The results are presented and analyzed in Chapter 5.

4.1. Sign Language Corpus and Labels

The first version of this dataset is presented in 2012 [44] which consisted of weather forecast interpretations of spoken German to German Sign Language (DGS), aired in a German public TV. The total duration of videos were three hours and signed by seven different signers. An extension is released in 2014 [45], in which the signer count became nine and total duration is tripled to 10 hours. It is split into sentences where sign glosses for each sentence are annotated by human experts. Transcriptions of accompanying spoken German narration are also available as well as hand shape and orientation annotations for some sentences in SignWriting [46] system. The final version of this dataset is presented in [1], with established recognition schemes for multi-signer (MS) and signer independent (SI) setups. The signer independent setup is split into training, development and test sets such that the test set is comprised of videos from an unseen signer (ID 05) and this signer is not included in training and development sets. Whereas the three subsets of the MS setup contain a mix of all nine signers. In our work, the MS setup is used for all of the experiments. Corpus statistics for different signer subsets are given in Table 4.1. All videos are recorded in a studio in

similar lighting conditions, have a resolution of 210 × 260 pixels, have constant frame rate of 25 frames per second and RGB modality only.

Table 4.1. Corpus statistics for training set of RWTH-PHOENIX-Weather Multi Signer dataset. A sign type is discoverable if it occurs two or more times.

Signer ID	Duration (min)	# Sentences	# Discoverable	
			Types	Tokens
1	130	1475	462	15928
5	125	1296	445	13795
4	82	836	345	7642
8	64	704	327	7242
7	60	646	390	7493
3	45	470	260	5227
9	17	165	203	1763
2	6	49	111	576
6	3	30	69	307
Total	533	5671	803	60927

In order to evaluate the performance of unsupervised term discovery algorithms, time interval of each gloss token needs to be labelled. Fortunately, these frame level labels are provided for training set of MS setup in Phoenix 2014 dataset [1]. The forced alignment procedure is used to obtain frame-level gloss annotations from sentence level manual gloss annotations. The *automatic* annotations are aligned by the method of Koller et al. [9] which employs a hybrid CNN-BLSTM-HMM model. The general training method for hybrid approach is described in Section 2.1.2. The gloss sequence w_1^N is initialized as flat-start for the HMM frame-state alignment, and re-aligned at each iteration of EM training. Each gloss is modelled by 3 state left-right HMM's. Garbage labels, similar to silence phones in ASR, are also employed to model movement epenthesis. Therefore the frame-level labels provided in Phoenix 2014 dataset contain garbage labels denoted as 'si' and the state index $s_t \in 0, 1, 2$ is appended to gloss labels,

such that WOLKE0, WOLKE1, WOLKE2 map to sign gloss WOLKE. Therefore we run our experiments on the training set of Phoenix 2014 MS setup [1] in order to make use of the automatically aligned labels for evaluation.

In addition to having time boundaries for labels, this corpus possesses other benefits for our task. The vocabulary is limited to weather related terms; there are only 1081 unique gloss types. Moreover, the signers are professionals which makes the inter-signer variability minimal.

4.2. Features

Among various visual features that are widely used for sign languages, we opt to utilize hand shapes and pose coordinates. Two set of high level visual features are used, which are extracted by running pre-trained models on each video frame.

4.2.1. Hand Shape Features

In 2016, Koller et al. [8] proposed a training procedure for training a CNN from weakly labelled data by employing the hybrid HMM-CNN training using iterative EM algorithm. They introduced the DeepHand convolutional network, which performs frame level classification between 60 different hand shapes from Danish taxonomy [47]. It is trained on about one million hand crops, belonging to three different sign language datasets: isolated Danish sign language [47] and New Zealand sign language [48] datasets, as well as a continuous RWTH Phoenix German sign language dataset [45].

The training procedure is initialized with flat start, meaning the sentence level annotations are distributed in equal intervals. The hand shape classes are modelled by left-to-right 3 state with 2 repetition HMM's, with global transition probabilities whereas the *garbage class* is modelled by an ergodic state to which transitions are possible from all other states.

The hand shape annotations for the RWTH Phoenix data [45] are derived from gloss annotations, by mapping glosses to corresponding entries in SignWriting [46] lexicon. The SignWriting lexicon is an open online database where users can add entries that link written language to sign notation consisting of hand shape, orientation and movement information.

The trained network is evaluated on two tasks, namely single frame hand shape recognition and continuous sign language recognition. The hand shape recognition performance is evaluated on the manually annotated subset of the RWTH Phoenix 2014 dataset [45]. The continuous sign recognition scheme is similar to [1] but the trained CNN is used as a feature extractor for the tandem HMM-GMM recognition model.

In our work, the DeepHand classifier is used as hand shape feature descriptor. For each video frame, we extract the 61 dimensional final layer activations before the softmax layer. We observed that reducing the dimensions to 40 by applying whitened PCA transformation improved the results in the discovery experiments. We only use the right hand features for all experiments.

4.2.2. Pose Features

Locations of skeleton joints are able to capture the form of signs. Therefore we use the joint coordinates for each frame as visual features. We use the 2D joint coordinates that are found by running OpenPose [49] estimator. We use the 25 keypoints pose output format for the main body joints and 21 keypoints for each hand. Concatenating the 8 upper body joints together with 21 keypoints for right and left hand each, we obtain 100 dimensional pose feature vectors per frame. The coordinates are normalized by subtracting the neck location and dividing by shoulder length.

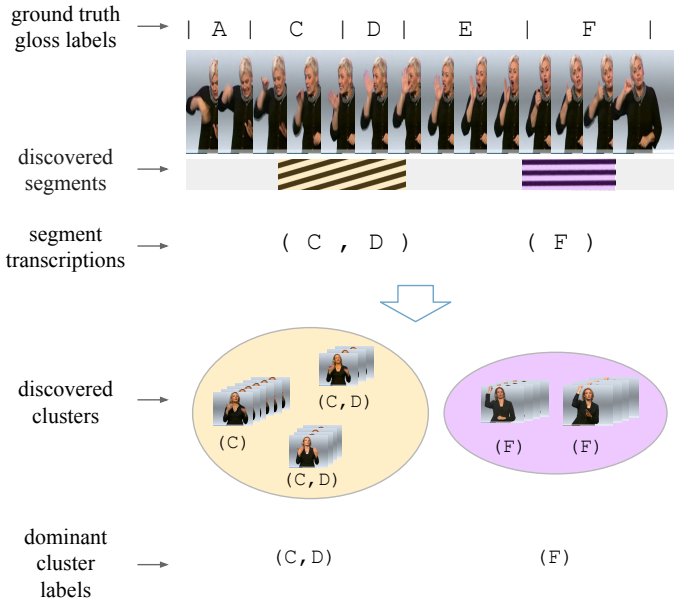


Figure 4.1. Illustration of evaluation scheme. The capital letters are hypothetical gloss labels.

4.3. Evaluation Metrics

The evaluation is performed with regard to the gold transcriptions of the discovered segments. For a segment $s_i = (f_i, m_i, n_i)$, the corresponding transcription $\tau(s_i)$ gives the gold unit labels from file f_i, which satisfy the overlapping conditions with the interval (m_i, n_i). In spoken UTD, the overlapping conditions are defined as 50% of phoneme duration or at least 30 ms [38]. For the sign language UTD, 50% condition may still be applicable but the 30 ms duration should be changed according to the expected sign unit length. For sign languages, we only apply the 50% condition, in order not to make assumptions on the expected term lengths. A simple visualization of the evaluation process is given in Figure 4.1. We modified the evaluation toolkit [50] provided for ZR speech challenges [34]. Some definitions that will be used in the evaluation measure descriptions are given below.

- The set of discovered clusters is defined as

$$C_{disc} = \{c_n |\ n \in \{1, .., N_{clus}\}\} \tag{4.1}$$

where N_{clus} is the total number of clusters and a single cluster c_n is composed of a set of discovered segments $\{s_i\}$.

- The set of non-overlapping discovered pairs that are clustered together is denoted by

$$P_{disc} = \{\{s_i, s_j\} \mid F_{olap}(s_i, s_j) = 0,\ s_i \in c,\ s_j \in c,\ c \in C_{disc}\} \tag{4.2}$$

where F_{olap} is defined in Eq. 3.10.

- The gold set of non-overlapping discovered pairs that can be created using the discovered segments is denoted by

$$\begin{aligned} P_{goldclus} = \{\{s_i, s_j\} \mid \exists c_1, c_2 \in C_{disc} : s_i \in c_1,\ s_j \in c_2, \\ \tau(s_i) = \tau(s_j),\ F_{olap}(s_i, s_j) = 0\}. \end{aligned} \tag{4.3}$$

- The set of individual segments belonging to a set of pairs is denoted by

$$\text{flat}(P) = \{s_i \mid \exists s_j : \{s_i, s_j\} \in P\}. \tag{4.4}$$

- The set of unique types that belong to a set of pairs is denoted by

$$\text{types}(P) = \{\tau(s) \mid s \in \text{flat}(P)\}. \tag{4.5}$$

- The set of segments that belong to a set of pairs P and have the same transcription t is denoted by

$$\text{match}(t, P) = \{s \in \text{flat}(P) \mid \tau(s) = t\}. \tag{4.6}$$

- The frequency of term t in set of pairs P is denoted by

$$\text{freq}(t, P) = \frac{|\text{match}(t, P)|}{|\text{flat}(P)|}. \tag{4.7}$$

4.3.1. Coverage

Coverage gives the fraction of input stream for which there are discovered segments. It does not take the quality of the clusters into account. In the original spoken UTD application [38], it is computed as the number of discovered phonemes over total number of phonemes in the corpus. The number of unique phonemes in a spoken language is around 50 for most of the worlds languages. For a long enough speech input, each of these phonemes occur many times, rendering the discovery of each possible. However for the sign languages, our target units are sign glosses and some of them may occur only once even in a large-scale dataset. Since a term must appear at least two times for it to become a motif, it would not be fair to expect 100% coverage for a sign UTD system. Therefore we modified the denominator of this metric such that only the *discoverable* terms are considered. Here, *discoverable* terms are the terms which occur more than once in the input stream. A formal definition of coverage metric is given as

$$\text{Coverage} = \frac{|\text{cover}(P_{disc})|}{|\text{cover}(P_{all})|} \tag{4.8}$$

where

$$\text{cover}(P) = \bigcup_{\langle m,n\rangle \in \text{flat}(P)} [m, n] \tag{4.9}$$

and P_{all} is the set of all discoverable pairs.

4.3.2. Normalized Edit Distance

Normalized edit distance (NED) is a metric that is used to evaluate the pairwise similarity of discovered tokens. It is based on the Levenstein distance between a pair

of strings, normalized to 0-1 interval. For a given pair, the Levenstein distance is the minimum edit distance between two sequence of discrete symbols, where the allowed editions are insertion, deletion and substitution. Then, the number of required edits are normalized by dividing the required number of edits by the length of the maximum length sequence. Finally, NED scores for all possible within-cluster pairs are averaged across all discovered pairs. Lower NED scores mean better matching. The formal definition can be made as

$$\text{NED} = \sum_{s_i, s_j \in P_{disc}} \frac{\text{ned}(s_i, s_j)}{|P_{disc}|} \tag{4.10}$$

where $\text{ned}(s_i, s_j)$ is the normalized Levenstein distance between two strings.

4.3.3. Grouping Quality

This set of metrics is computed in terms of precision (P), recall (R) and F-score (harmonic mean of P and R). It is similar to cluster purity and inverse purity. If the pairs within a cluster has the same transcription, then the precision is high. If pairs from separate clusters have the same transcriptions, then the recall is low. The grouping precision and recall are defined as

$$\text{Grouping P.} = \sum_{t \in \text{types}(\text{flat}(P_{disc}))} \text{freq}(t, P_{disc}) \ \frac{|\text{match}(t, P_{disc} \cap P_{goldclus})|}{|\text{match}(t, P_{disc})|} \tag{4.11}$$

$$\text{Grouping R.} = \sum_{t \in \text{types}(\text{flat}(P_{goldclus}))} \text{freq}(t, P_{goldclus}) \ \frac{|\text{match}(t, P_{disc} \cap P_{goldclus})|}{|\text{match}(t, P_{goldclus})|} \tag{4.12}$$

and then

$$\text{F-score} = \frac{2}{1/\text{Precision} + 1/\text{Recall}} \tag{4.13}$$

is their harmonic mean.

4.4. Experiment Pipeline

4.4.1. Adjusting Cost Threshold

The cost threshold θ enables us to adjust the trade-off between coverage and matching quality. NED score is used to assess matching quality and typically NED increases as the coverage increases. In order to achieve fair comparison to spoken term discovery results, we aimed around 10% for coverage values. However, the appropriate threshold that achieves 10% coverage varies, according to the parameter combination. Therefore the cost threshold is swept until the Coverage value is between 9% and 11%.

4.4.2. Cross Validation Scheme

The dataset is partitioned into three folds for cross-validation, as shown in Table 4.2. The sizes in terms of number of sentences are matched while partitioning into three subsets. At each fold, 1/3 of the data is used as development set and the remaining is used as unseen test set. The best parameter combinations are found for the development sets as described in Section 4.4.3, and those combinations are used for the unseen test sets. Then, the development is changed and the tuning procedure is re-run. The results are then reported using the average of test results, weighted by number of sentences. For each experiment, the final cost threshold θ is adjusted so that Coverage is about 10%, and NED score is used as decision criterion. The signers are distributed into separate development and test partitions in order to observe the discovery performance when the algorithms are tested on new signers.

4.4.3. Hyper-Parameter Optimization

Out of the three discovery algorithms that we use, each has a set of hyper-parameters that need to be optimized. However not all hyper-parameters affect the results significantly. Therefore for each algorithm, the effect of individual parameters on the matching performance is assessed by sweeping the value of one parameter while

Table 4.2. Partitions of the Phoenix 2014 MS [1] for cross-validated experiments

Subsets	Signer IDs	# Sentences	Total Size
1	4	836	1705
	8	704	
	9	165	
2	1	1475	1975
	3	470	
	6	30	
3	5	1296	1991
	7	646	
	2	49	

fixing the rest. Then, only the parameters which influence the results are considered for tuning at the development set. A Bayesian optimization procedure named Gaussian process regression is applied, where the cost function is approximated by a Gaussian process. The function values are assumed to be drawn from a multivariate Gaussian. The cost value to be minimized is chosen to be NED score at 10% coverage. The Gaussian process regression method allows us to make as few assumptions as possible about the nature of the cost function.

5. EXPERIMENTS AND RESULTS

The results that are presented in this chapter are obtained using three different UTD algorithms, which are described in detail in Chapter 3. All three of these algorithms share a common pipeline for discovery and evaluation. This pipeline is explained in Section 4.4. The first algorithm that we experimented with is the segmental DTW (SDTW) based algorithm (see Section 3.1.1), which is the pioneering spoken UTD algorithm. This SDTW based algorithm is very slow, therefore the results presented in Section 5.1 are not comprehensive but rather preliminary results. Instead, more comprehensive experiments are carried out using the efficient SDTW (E-SDTW) [5] algorithm and the KNN based [6] algorithm. The matching performances of these two algorithms are compared. Moreover, two different clustering strategies are applied for KNN and E-SDTW based matches. The examples of discovered clusters are displayed in Appendix A. For all setups, nearly 1% of the perfect matches come from different signers, most of the correct matches belong to the same signer. Therefore we can think of these results as the average of signer dependent experiments.

5.1. Segmental DTW Based Experiments

The effect of each parameter on the matching performance is assessed by sweeping the value of one parameter while fixing the rest. Cosine distance is used as the distance metric between hand shape feature vectors. We fixed the minimum path length L_{min} to 6 frames, the width of diagonal search band R to 8 frames and extension ratio $extend_{ratio}$ to 1 of the LCMA length. Initial experiments showed that the width of the diagonal band R did not have much effect on the results, but setting a larger R decreased the number of computations. Therefore we fixed $R = 20$ frames for the rest of the experiments. For the remaining parameters, the optimization procedure in Section 4.4.3 is followed. However, due to the time complexity of the SDTW algorithm, which is $O(n^2)$, it is extremely slow. Moreover, due to the fact that each input file is compared to all other input files, the computation time is also quadratic as a function

of number of input sequences. Therefore, we did not carry out the experiments on the entire partitions but instead divided each partition into 5 subsets and used the averaged results.

Table 5.1. Discovery results using the SDTW algorithm and hand shape features, at 10% Coverage.

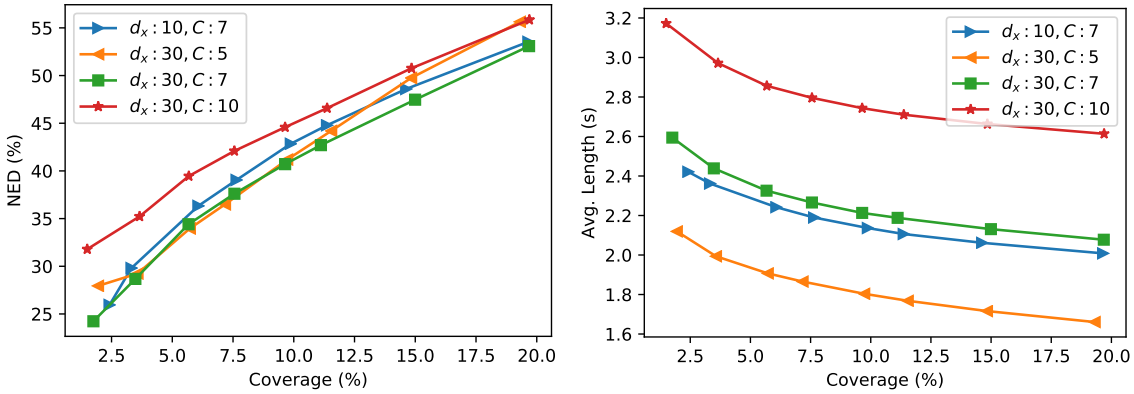
Clustering	NED	Grouping			Avg. Seg.	# Discovered	
	(%)	P (%)	R (%)	F (%)	Length (sec)	Clusters	Segments
Pairs	**51.0**	54.8	53.9	54.2	0.43	303.5	607.1
Word Centers	75.3	**71.0**	**82.0**	**76.0**	0.40	53.5	351.3

The results for SDTW experiments with hand shape features are shown in Table 5.1. The clustering method that is applied here is the same with the original work [4]. These results are not directly comparable to the results of the other two algorithms because here we average the results of smaller subsets due to long computation times. Nevertheless, we can observe that the matching phase discovers meaningful pairs, and clustering stage yields high quality clusters.

5.2. Efficient SDTW Baseline Algorithm

For the E-SDTW algorithm, we modified the baseline software provided for Zero Resource challenges [2] so that it runs on the extracted visual features. The procedure described in Section 4.4.3 is carried out to determine the optimum combination of parameters for each development set. Even though the optimum values of parameters for each development set vary, they are not much different from each other. Among the default parameters for spoken UTD, the parameters that needed to be optimized together with their optimum values are $\delta = 0.35,\ d_x = 30,\ \mu = 0.45$ and $C = 7$. The effect of changing C and d_x on matching performance and average segment lengths are shown in Figure 5.1. The C parameter determines the threshold to terminate the SDTW search, thus increasing C yields longer segments. Similarly, d_x is the length of diagonal $\mu - percentile$ filter, thus increasing d_x enables longer diagonals to be filtered

out of the sparse similarity matrix.



(a) NED scores

(b) Average lengths

Figure 5.1. NED and average segment length vs. Coverage curves for various values of the parameters C and d_x for the E-SDTW (DH) setup.

The relation between match similarity and average length of segment pairs is shown in Figure 5.2. Here, the cost values relate to the DTW alignment costs that are normalized by path lengths. It is easy to observe that as the segment lengths increase, the matching costs decrease. This might be due to the implementation of path normalization in ZRTools software [51], such that warping cost is divided by $l_1 + l_2$ instead of the total path length.

As mentioned earlier, the desired coverage is obtained by adjusting the cost threshold. Since lower cost pairs have longer segment lengths, as coverage increases, shorter segment pairs are also included. This statement can be verified by observing Figure 5.3, where the pairwise matches at different coverage values are displayed. The axes are the lengths of each segment of a pair, and the colors indicate the average purity of all matches that fall into corresponding bins of lengths. Another observation that can be made is that as the segment length ratio of a pair diverges from unity, the matches tend to become less pure.

The results using hand shape features are displayed in Table 5.2. These results are obtained by optimizing the parameters for NED scores at 10 % coverage of the

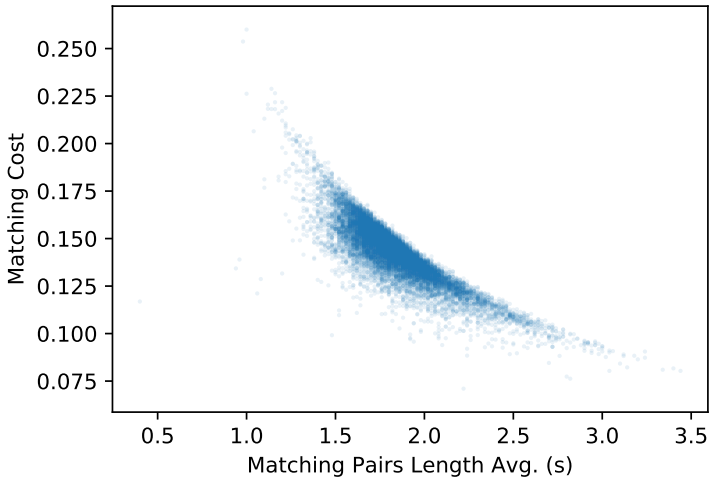
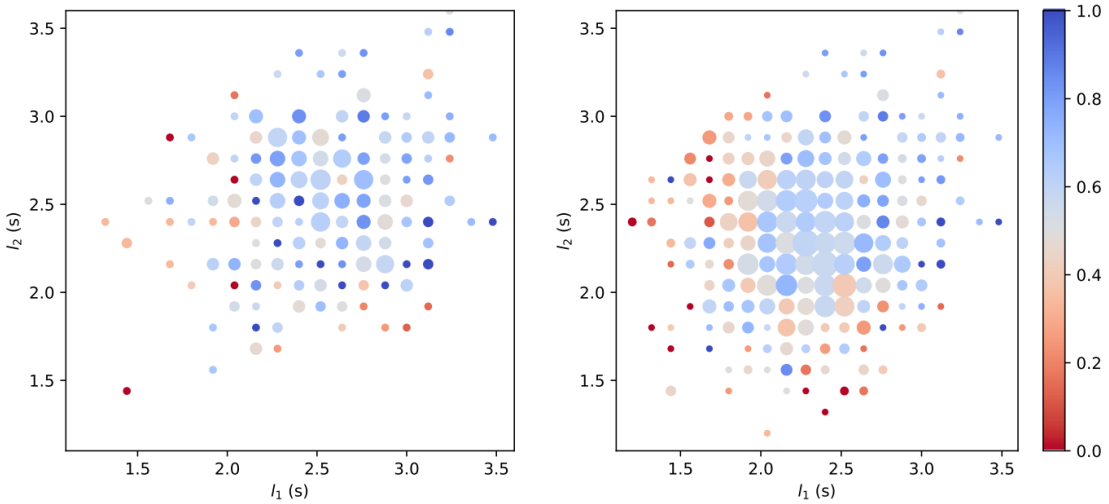


Figure 5.2. The relation between the DTW alignment costs and average pair lengths, for E-SDTW algorithm and hand shape features.



(a) 5% Coverage (b) 10% Coverage

Figure 5.3. The segment lengths vs purity of matching pairs for E-SDTW (DH) setup, where the colors indicate matching purity and the sizes are proportional to the total number of pairs at those lengths.

matching stage. Then, the two clustering methods are run using these matches. The clustering method (SC) proposed by Park and Glass [4] performs better compared to the overlap edges (OE) based method, in terms of both grouping precision and recall. We can observe that SC clustering achieves the same coverage with less segments. This is due to the fact that segments are not allowed to overlap with another segment from the same cluster but are allowed to overlap with segments from another cluster. Since there are more segments per cluster with the SC method, the amount of overlap is less compared to OE method.

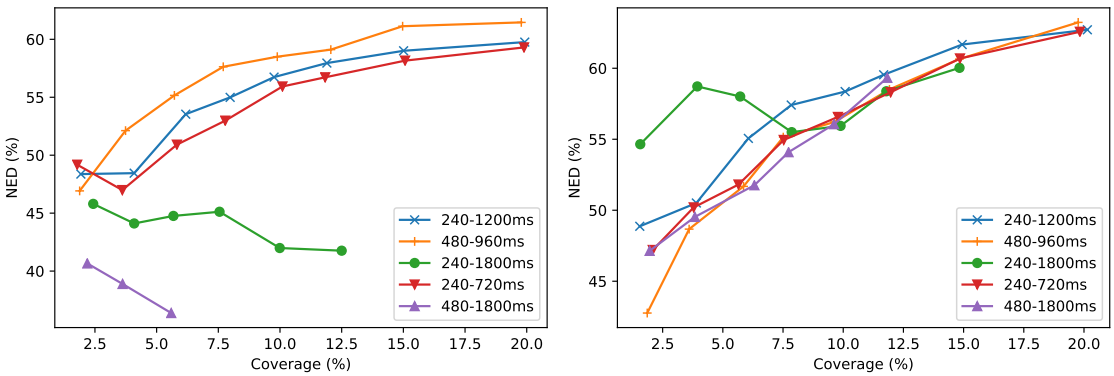
Table 5.2. Discovery results using the Efficient SDTW algorithm and hand shape features, at 10% Coverage.

Phase	Set	NED	Grouping			Avg. Seg.	# Discovered	
		(%)	P (%)	R (%)	F (%)	Length (sec)	Clusters	Segments
Matching	Dev	**39.1**	19.6	51.7	28.1	2.12	1017.3	2034.6
	Test	41.0	18.9	51.8	27.4	2.10	994.9	1989.7
Clustering (SC) (Modularity)	Dev	78.4	20.5	88.0	32.9	1.95	99.2	611.7
	Test	74.1	**22.3**	**88.9**	**35.3**	1.96	94.1	576.7
Clustering (OE) (C.Comp)	Dev	75.8	17.5	76.2	28.0	2.04	213.2	730.8
	Test	72.8	19.3	80.9	30.8	2.02	208.6	738.8

5.3. KNN Based Discovery

We first explored the effect of different hyper-parameters on discovery performance. The expected term length for the Phoenix dataset [1] is about 10 frames (0.4s). Using this information, we set the minimum segment length l_{min} as 6 frames (240ms) and segmentation resolution a to be 3 frames. We observed that increasing the maximum segment length l_{max} to 45 frames (1.8s) allowed discovery of n-grams. We tried different segment length constraints using both types of features, in order to observe whether the type of feature affects the optimum interval. The NED - Coverage curves that resulted from these experiments are shown in Figure 5.4. Here, interest-

ingly, the hand shape features behave differently for longer segments. For the hand shape features, the NED scores improve as the cost threshold is increased. This implies that there is a discrepancy between the NED scores and matching costs, such that the embeddings for hand shape features of high quality long matches have somehow high cost and thus not included in low coverage settings.

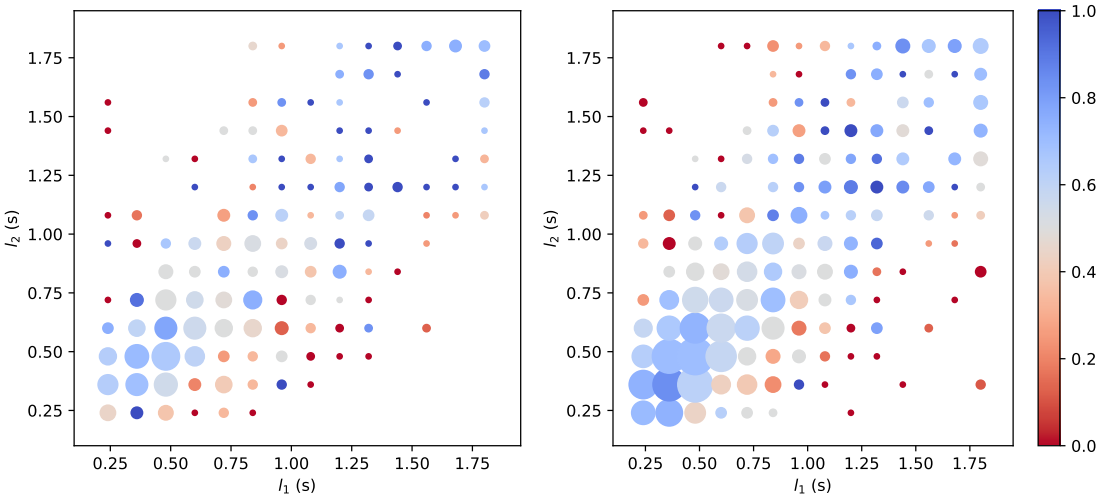


(a) KNN (DH) (b) KNN (OP)

Figure 5.4. The effect of different segment intervals (l_{min}, l_{max}) on matching performance, for hand shape (DH) and pose (OP) features.

With a, l_{min} and l_{max} fixed, we then perform cross-validated grid search to find the best combination of embedding dimension L and smoothing parameters r and s. Even though the optimum values for these parameters vary for each signer and type of feature, we observed that setting $r = 0.1, s = 0.5$ and $L = 6$ frames yield good results in general.

The distribution of matches with respect to both segment lengths are displayed in Figure 5.5. Here, the colors indicate the purity of the matches and the marker sizes indicate the number of matches having those lengths. We can observe that increasing the coverage does not affect the segment lengths significantly, as opposed to the case for E-SDTW experiments (see Figure 5.3). We can also observe that as the ratio between lengths of segments diverge from unity, the pairs become less pure.



(a) 5% Coverage

(b) 10% Coverage

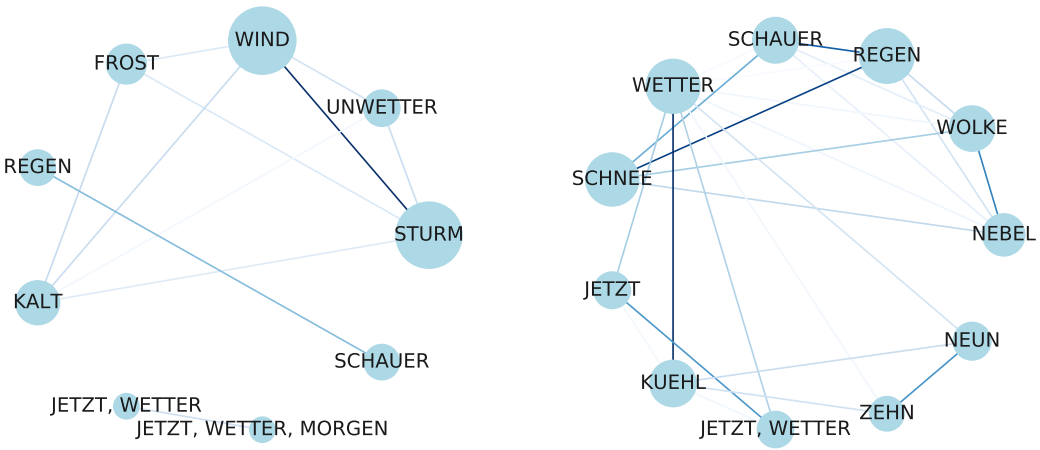
Figure 5.5. The segment lengths vs purity of matching pairs for KNN (DH) setup, where the colors indicate matching purity and the sizes are proportional to number of pairs.

Among the two sets of features (DeepHand and OpenPose [8, 49]), the DeepHand features yield better results as shown in Table 5.3. It can also be observed that SC Clustering method performs better in grouping quality metrics, with higher segment per cluster ratio.

Table 5.3. Discovery results using the KNN based algorithm for both hand shape (DH) and pose (OP) features, at 10% Coverage.

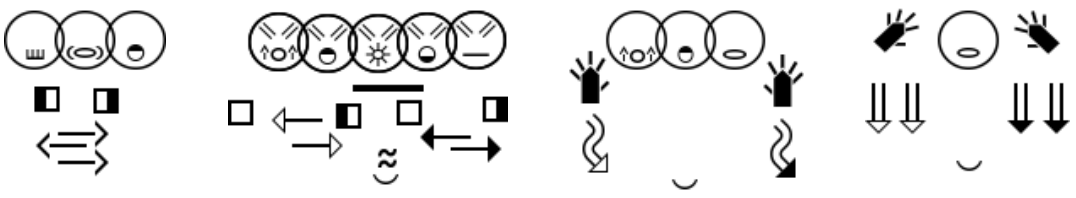
Feature	Phase	Set	NED	Grouping			Avg. Seg.	# Discovered	
			(%)	P (%)	R (%)	F (%)	Length (sec)	Clusters	Segments
DH	Matching	Dev	41.4	50.8	52.8	51.8	0.51	1279.9	2559.8
		Test	43.4	50.1	52.0	51.0	0.52	1359.2	2718.4
	Clustering (SC) (Modularity)	Dev	79.1	72.5	84.3	77.9	0.56	135.5	1240.7
		Test	85.6	72.9	84.0	78.0	0.57	139.4	1199.0
	Clustering (OE) (C.Comp)	Dev	76.5	59.7	66.3	62.8	0.51	599.6	1770.6
		Test	82.8	59.4	65.3	62.1	0.52	599.6	1751.4
OP	Matching	Dev	48.6	43.7	40.0	41.5	0.36	1090.6	2181.1
		Test	50.7	43.1	39.7	41.0	0.37	1206.4	2412.8
	Clustering (SC) (Modularity)	Dev	88.7	60.4	72.0	65.4	0.44	167.5	1337.1
		Test	89.6	61.0	72.1	65.9	0.45	158.1	1352.2
	Clustering (OE) (C.Comp)	Dev	87.3	49.3	54.3	51.4	0.37	627.3	1959.5
		Test	82.6	49.2	53.8	51.3	0.38	590.6	1814.1

The most confused pairs for each type of feature are given in Figure 5.6. Here, we observe that semantically similar signs (e.g. rain-shower, wind-storm etc.) which also have similar forms (see Figure 5.7) are easily confused. The SignWriting notations of wind (WIND) and storm (STURM) glosses have similar hand shapes but the movement patterns are different. Thus it is not surprising to have them confused using the hand shape features. Interestingly, the most confused glosses differ according to feature type. This observation leads to the conclusion that in future studies, these two types of features can be fused together to complement each others weaknesses.



(a) KNN (DH) (b) KNN (OP)

Figure 5.6. Examples of the most confused pairs of glosses for hand shape (DH) and pose (OP) features. Darker lines represent more confused pairs and circle radii are proportional to discovered gloss frequencies.

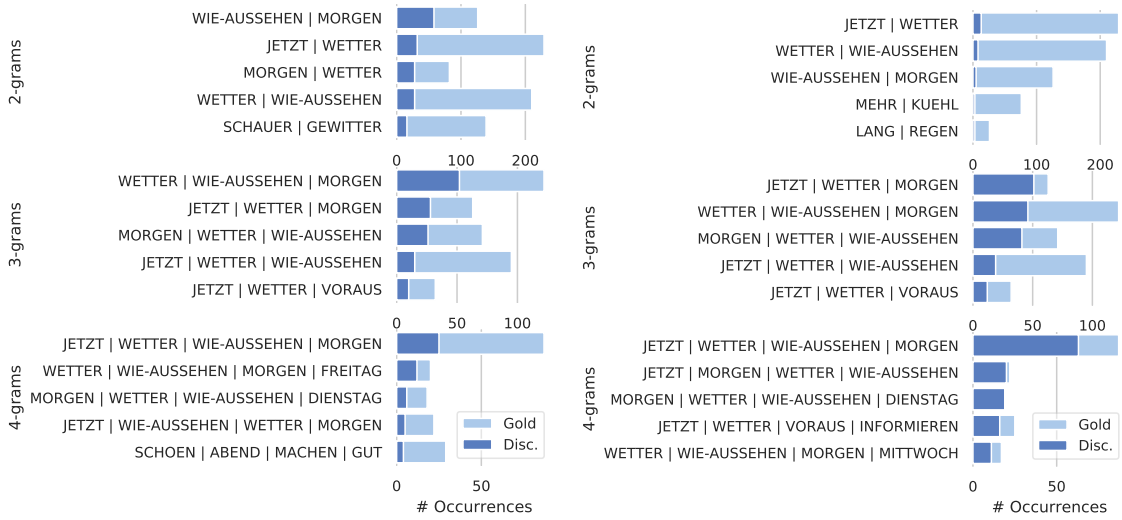


(a) WIND (b) STURM (c) SCHNEE (d) REGEN

Figure 5.7. SignWriting entries of some of the most confused gloss pairs. Notice that a-b and c-d have similar signings.

5.4. Comparison and Fusion of E-SDTW and KNN Experiments

The most important difference between the KNN based algorithm [6] and E-SDTW baseline [5] is the length of discovered segments. The E-SDTW baseline is able to discover longer segments, often n-grams, therefore fewer pairs are needed to satisfy 10% Coverage. Correctly discovered n-grams for both algorithms are displayed in Figure 5.8.



(a) KNN (DH)

(b) E-SDTW (DH)

Figure 5.8. Examples of correctly discovered gloss n-grams for both algorithms, together with the number of occurrences of existing gold n-grams

The reason for the difference in segment duration might be explained as follows. The E-SDTW algorithm uses a sparse approximation of the distance matrix. Then, the SDTW search is performed on the diagonal segments that remain after the median filtering of the sparse matrix. Median filtering allows only long diagonal paths to be passed. The elements of short diagonals might not be abundant enough in the sparse matrix to be filtered as diagonal. Therefore the segments with this method tend to be longer. If videos with higher sampling rate are used or the degree of approximation (thus amount of speed-up) is decreased, shorter segments may also be discovered. Conversely, KNN based algorithm is more receptive to shorter segments, because smoothed embedding method may cause less distortion for shorter segments. More complex fixed-

length representation methods may preserve the information spanning longer segments.

As shown in Table 5.3 and Table 5.2, NED scores for both algorithms are similar. However, grouping precision of the KNN based algorithm is considerably better. Another notable difference is that, using pose features does not significantly degrade discovery performance for the KNN based algorithm, whereas we were not able to obtain good results with pose features and E-SDTW setup. It should also be noted that KNN based algorithm runs much faster; pre-computed features for 3 hours of video is processed in about one minute using GPU, versus 10 minutes using the SDTW baseline.

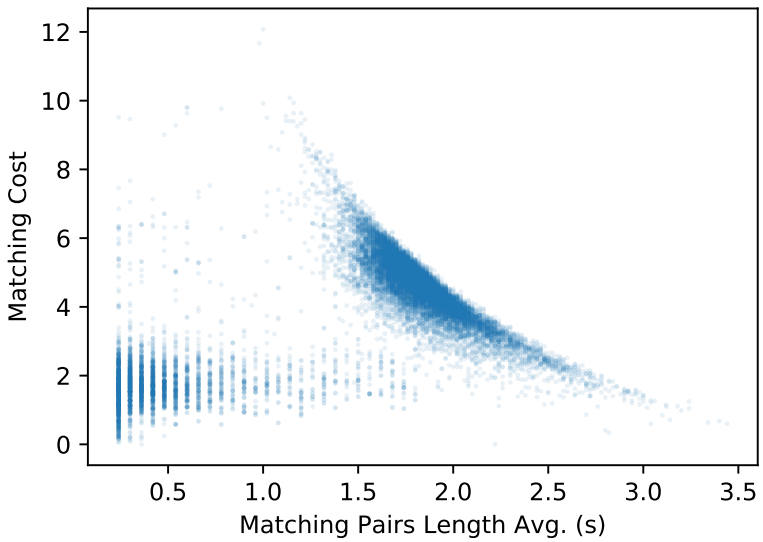


Figure 5.9. The relation between the normalized matching costs and average pair lengths, for the fusion of KNN (DH) and E-SDTW (DH).

The E-SDTW and KNN based algorithms are better for different duration matches. Therefore the results can be fused together to complement each other's weaknesses. The fusion strategy that we followed is to simply join the two set of matching pairs. In order to adjust the coverage amount, the matching costs of the two algorithms should be close, so that there are matches from both sets below a certain cost threshold. However, the matches are scored differently for each algorithm. Therefore we apply a simple cost normalization such that they have unit variance and the minimum costs for

both are zero. The distribution of joined matches with respect to average duration and matching costs are shown in Figure 5.9. Here, we can observe two separate clusters, where the cluster on the right comes from the E-SDTW method (see Figure 5.2) and the other cluster belongs to KNN matches.

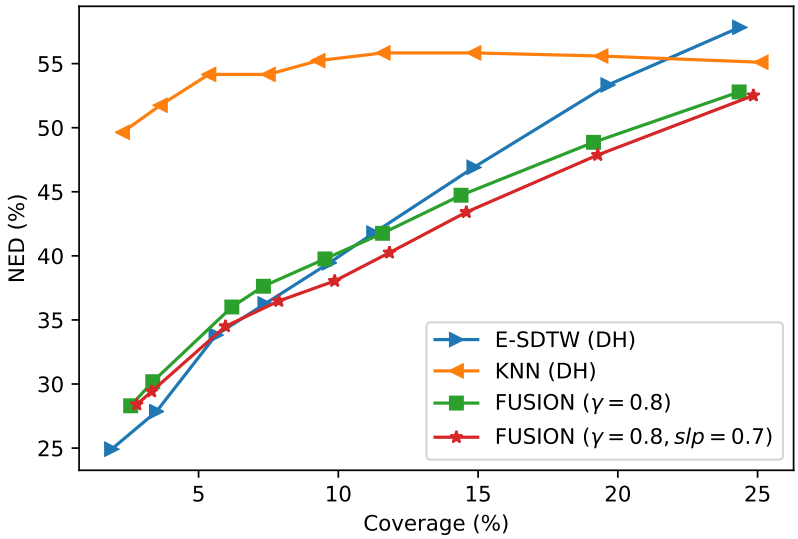
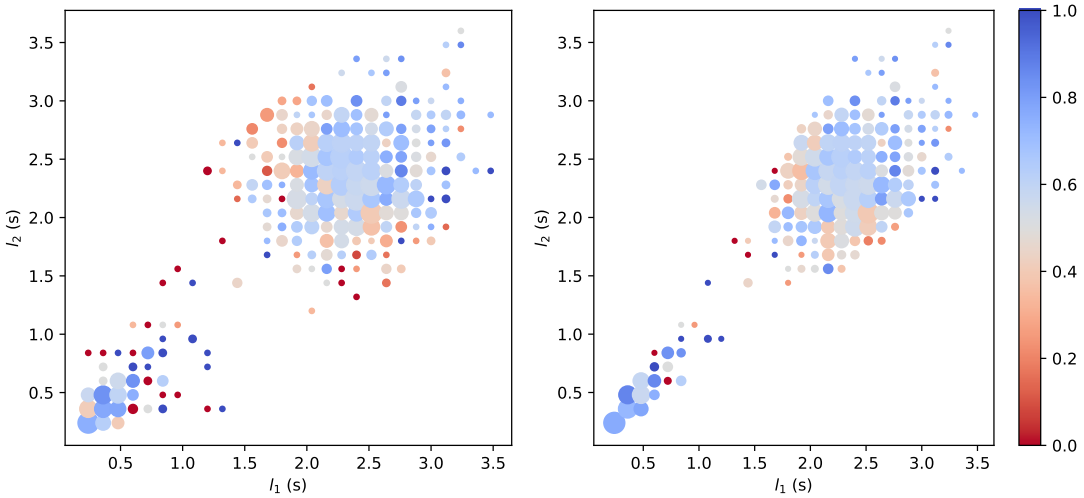


Figure 5.10. NED - Coverage curves for the fusion of KNN (DH) and E-SDTW (DH).

The number of matches from both methods that are below a certain threshold may be much different and this can worsen the results. Thus, we scale the costs of E-SDTW matches by a factor γ so that the mixing ratio can be adjusted. The optimum γ is adjusted by observing NED - Coverage curves. The optimum fusion curve together with the curves for KNN and E-SDTW matches are shown in Figure 5.10. The results are given as curves instead of being fixed at 10% Coverage in order to better observe the fusion characteristics at different regions.

The distribution of fusion matches with respect to both segment lengths are displayed in Figure 5.11 on the left. Again, we observe that as the duration of segment pairs differ from each other, the matching quality worsens. In order to combat this effect, we apply another selection criterion such that we discard the matches whose ratio of segment durations (shorter / longer) are below a certain threshold. The matches after applying the duration ratio threshold of 0.7 are shown in Figure 5.11 on the right. It is



(a) Fusion

(b) Fusion with length ratio constraint

Figure 5.11. The segment lengths vs purity of pairs for the fusion of KNN (DH) and E-SDTW (DH) matches at 10% Coverage, where the colors indicate purity.

clear that higher quality matches remain and impure matches are discarded. The effect of applying duration ratio threshold can also be observed from the NED - Coverage curves in Figure 5.10, where the improvement is evident at all coverage ranges.

6. CONCLUSION AND FUTURE WORK

This thesis is presented as an initial exploration of unsupervised term discovery methods for continuous sign language. The main goal is to show that spoken term discovery algorithms can be applied on sign languages by using visual features. We used two types of features, hand shapes and pose keypoints. We obtained the features for each frame by running pre-trained models on RGB videos from the Phoenix Weather 2014 continuous sign dataset. These features are then fed to the three different term discovery algorithms all of which return pairs of similar segments. We also applied two different graph clustering strategies to obtain clusters of hypothesized words. The gloss annotations of the dataset, for which time boundaries were given, enabled us to evaluate and compare the performance of different discovery setups.

We first demonstrate that the pioneering spoken UTD algorithm, which employs segmental DTW, is able to discover sign terms. However, this algorithm is very slow and we could not run comprehensive experiments to compare with other algorithms. Fortunately, we use a faster version of this algorithm, referred as Efficient-SDTW, and we were able to obtain successful results using hand shape features. A KNN based spoken term discovery algorithm is also implemented, which performs similarly to E-SDTW algorithm and is much faster than E-SDTW method. Moreover, we were able to get successful results using pose features with this algorithm. We show that the E-SDTW method is better at discovering longer sequences while KNN method is better for discovering shorter segments. A fusion of matching pairs from these two algorithms is shown to improve results by combining both short and long matches. We also observed that the clustering method that finds and clusters possible word centers is more successful in terms of grouping quality metrics. Experiments also show that the UTD algorithms perform comparable to spoken term discovery counterparts, in terms of coverage, NED and grouping quality metrics.

For the scope of this study, we did not focus on finding the best feature extraction methods for sign terms discovery. Instead, we primarily relied on the DeepHand features, which is in fact against the zero-resource constraint in the sense that the training procedure of DeepHand network utilizes the hand shape information of the Phoenix dataset. Our aim is to demonstrate the applicability of UTD algorithms given powerful visual features (such as MFCC for speech). Nevertheless, we also showed that UTD can also be successful using OpenPose features, which obey the ZR constraints since its training is independent of sign language recognition tasks.

An area for future investigation is to learn better feature representations that will be robust to non-linguistic variances such as signing speed, illumination differences, camera angles etc. Using the discovered pairs, representation learning methods such as frame-wise correspondence autoencoders (CAE) or sequence to sequence CAE's can be used. The KNN based method is flexible in the sense that, more sophisticated segmentation and embedding approaches can be incorporated easily. Henceforth, a future direction is to improve on fixed-length representation learning methods, which may also combine non-manual modalities.

The proposed approach can aid sign language community in numerous ways. First of all, it can be very useful in cases where there are large amounts of sign videos to be annotated but not enough available resources. The algorithm proposes segments and clusters them so that each cluster corresponds to a hypothesized gloss. An educated annotator can easily purify the discovered clusters by eliminating the false segments and then, saving the segments from pure clusters as annotations. By doing so, a significant amount of data can be annotated in short time.

We applied the same evaluation metrics with spoken UTD, however, we used gloss transcriptions instead of phonemes. The discovered segments in spoken UTD usually cover at least three phonemes, therefore using NED as a criterion for matching quality is more meaningful. In our application, most of the discovered segments correspond to one or more glosses, therefore NED scores are influenced more if pairs are partially

different. Moreover, In future studies, these metrics can be tailored for the type of labels that are available. Also, area under the NED - Coverage curves at a certain range can be minimized to compare different setups and run parameter tuning. The cross-validated evaluation scheme that we propose may be used in future studies to benchmark other UTD algorithms and representation learning methods.

The quality of the segmentation boundaries is not assessed in this work. In a future study, a psycho-linguistic experiment can be carried out, where the subjects are shown signs that are segmented by humans versus the UTD algorithm and they are asked to decide on which segmentation seems more natural. This would validate the potential use case of the UTD method as an automatic segmentation tool. Another benefit may arise when we want to train an ASLR system on a sign language that does not have enough resources. The clusters found by the UTD algorithm can provide weak supervision for model training.

One of the drawbacks of this work is having used only one corpus for development and testing. Although the testing is done on unseen signers, the language is the same and the recording conditions are almost identical. A future study should include another sign corpus with a different sign language for testing. This would enforce the system to be language independent and would require better feature representations that can generalize well.

REFERENCES

1. Koller, O., J. Forster and H. Ney, "Continuous sign language recognition: Towards large vocabulary statistical recognition systems handling multiple signers", *Computer Vision and Image Understanding*, Vol. 141, pp. 108–125, 2015.

2. Versteegh, M., R. Thiollière, T. Schatz, X.-N. Cao Kam, X. Anguera, A. Jansen and E. Dupoux, "The Zero Resource Speech Challenge 2015", *Proc. Interspeech*, pp. 3169–3173, 2015.

3. Dunbar, E., X. Cao, J. Benjumea, J. Karadayi, M. Bernard, L. Besacier, X. Anguera and E. Dupoux, "The zero resource speech challenge 2017", *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 323–330, 2017.

4. Park, A. S. and J. R. Glass, "Unsupervised Pattern Discovery in Speech", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 16, No. 1, pp. 186–197, 2008.

5. Jansen, A. and B. V. Durme, "Efficient spoken term discovery using randomized algorithms", *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 401–406, 2011.

6. Thual, A., C. Dancette, J. Karadayi, J. Benjumea and E. Dupoux, "A K-Nearest Neighbours Approach To Unsupervised Spoken Term Discovery", *Proc. 2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 491–497, 2018.

7. Polat, K. and M. Saraçlar, "Unsupervised Term Discovery for Continuous Sign Language", *Proceedings of the LREC2020 9th Workshop on the Representation and Processing of Sign Languages: Sign Language Resources in the Service of the Language Community, Technological Challenges and Application Perspectives*, pp.

189–196, European Language Resources Association (ELRA), Marseille, France, 2020.

8. Koller, O., H. Ney and R. Bowden, "Deep Hand: How to Train a CNN on 1 Million Hand Images When Your Data Is Continuous and Weakly Labelled", *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3793–3802, 2016.

9. Koller, O., S. Zargaran and H. Ney, "Re-Sign: Re-Aligned End-to-End Sequence Modelling with Deep Recurrent CNN-HMMs", *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3416–3424, 2017.

10. Pfister, T., J. Charles and A. Zisserman, "Large-scale learning of sign language by watching TV (using co-occurrences)", *Proceedings of the British Machine Vision Conference*, pp. 1–11, 2013.

11. Buehler, P., A. Zisserman and M. Everingham, "Learning sign language by watching TV (using weakly aligned subtitles)", *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2961–2968, 2009.

12. Kelly, D., J. Mc Donald and C. Markham, "Weakly Supervised Training of a Sign Language Recognition System Using Multiple Instance Learning Density Matrices", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 41, No. 2, pp. 526–541, 2011.

13. Nayak, S., S. Sarkar and B. Loeding, "Unsupervised Modeling of Signs Embedded in Continuous Sentences", *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) - Workshops*, pp. 81–81, 2005.

14. Nayak, S., K. Duncan, S. Sarkar and B. L. Loeding, "Finding Recurrent Patterns from Continuous Sign Language Sentences for Automated Extraction of Signs", *Journal of Machine Learning Research*, Vol. 13, pp. 2589–2615, 2012.

15. Theodorakis, S., V. Pitsikalis and P. Maragos, "Dynamic-static unsupervised sequentiality, statistical subunits and lexicon for sign language recognition", *Image Vision Comput.*, Vol. 32, pp. 533–549, 2014.

16. Yin, P., T. Starner, H. Hamilton, I. Essa and J. M. Rehg, "Learning the basic units in American Sign Language using discriminative segmental feature selection", *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4757–4760, 2009.

17. Tornay, S. and M. Magimai.-Doss, "Subunits Inference and Lexicon Development Based on Pairwise Comparison of Utterances and Signs", *Information*, Vol. 10, p. 298, 2019.

18. Siyli, R. D., B. Gundogdu, M. Saraclar and L. Akarun, "Unsupervised Key Hand Shape Discovery of Sign Language Videos with Correspondence Sparse Autoencoders", *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8179–8183, 2020.

19. Renshaw, D., H. Kamper, A. Jansen and S. Goldwater, "A comparison of neural network methods for unsupervised representation learning on the zero resource speech challenge", *Sixteenth Annual Conference of the International Speech Communication Association*, pp. 3199–3203, 2015.

20. Kamper, H., M. Elsner, A. Jansen and S. Goldwater, "Unsupervised neural network based feature extraction using weak top-down constraints", *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5818–5822, 2015.

21. Synnaeve, G., T. Schatz and E. Dupoux, "Phonetics embedding learning with side information", *Proc. IEEE Spoken Language Technology Workshop (SLT)*, pp. 106–111, 2014.

22. Last, P., H. A. Engelbrecht and H. Kamper, "Unsupervised Feature Learning for Speech Using Correspondence and Siamese Networks", *IEEE Signal Process. Lett.*, Vol. 27, pp. 421–425, 2020.

23. Bhati, S., S. Nayak, K. S. R. Murty and N. Dehak, "Unsupervised Acoustic Segmentation and Clustering Using Siamese Network Embeddings.", *Proc. Interspeech*, pp. 2668–2672, 2019.

24. Levin, K., K. Henry, A. Jansen and K. Livescu, "Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings", *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 410–415, 2013.

25. Holzenberger, N., M. Du, J. Karadayi, R. Riad and E. Dupoux, "Learning Word Embeddings: Unsupervised Methods for Fixed-size Representations of Variable-length Speech Segments", *Proc. Interspeech*, pp. 2683–2687, 2018.

26. Kamper, H., "Truly Unsupervised Acoustic Word Embeddings Using Weak Top-down Constraints in Encoder-decoder Models", *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6535–3539, 2019.

27. Park, A. and J. R. Glass, "Towards unsupervised pattern discovery in speech", *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 53–58, 2005.

28. Muscariello, A., G. Gravier and F. Bimbot, "Unsupervised Motif Acquisition in Speech via Seeded Discovery and Template Matching Combination", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 20, No. 7, pp. 2031–2044, 2012.

29. Lee, C.-y. and J. Glass, "A Nonparametric Bayesian Approach to Acoustic Model Discovery", *Proceedings of the 50th Annual Meeting of the Association for Com-*

putational Linguistics (Volume 1: Long Papers), pp. 40–49, Jeju Island, Korea, 2012.

30. Teh, Y., M. Jordan, M. Beal and D. Blei, "Hierarchical Dirichlet Processes", *Machine Learning*, pp. 1–30, 2006.

31. Heck, M., S. Sakti and S. Nakamura, "Unsupervised Linear Discriminant Analysis for Supporting DPGMM Clustering in the Zero Resource Scenario", *Procedia Computer Science*, Vol. 81, pp. 73 – 79, 2016.

32. Ondel, L., L. Burget and J. Černocký, "Variational Inference for Acoustic Unit Discovery", *Procedia Computer Science*, Vol. 81, pp. 80–86, 2016.

33. Blei, D. and M. Jordan, "Variational inference for Dirichlet process mixtures", *Bayesian Analysis*, Vol. 1, pp. 121–144, 2006.

34. Dunbar, E., R. Algayres, J. Karadayi, M. Bernard, J. Benjumea, X.-N. Cao, L. Miskic, C. Dugrain, L. Ondel, A. W. Black, L. Besacier, S. Sakti and E. Dupoux, "The Zero Resource Speech Challenge 2019: TTS without T", *Proc Interspeech*, pp. 1088–1092, Graz, Austria, 2019.

35. Ebbers, J., J. Heymann, L. Drude, T. Glarner, R. Haeb-Umbach and B. Raj, "Hidden Markov Model Variational Autoencoder for Acoustic Unit Discovery.", *Proc. Interspeech*, pp. 488–492, 2017.

36. Ondel, L., P. Godard, L. Besacier, E. Larsen, M. Hasegawa-Johnson, O. Scharenborg, E. Dupoux, L. Burget, F. Yvon and S. Khudanpur, "Bayesian models for unit discovery on a very low resource language", *Proc IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5939–5943, 2018.

37. Kamper, H., A. Jansen and S. Goldwater, "A segmental framework for fully-unsupervised large-vocabulary speech recognition", *Computer Speech & Language*, Vol. 46, p. 154–174, 2017.

38. Ludusan, B., M. Versteegh, A. Jansen, G. Gravier, X.-N. Cao, M. Johnson and E. Dupoux, "Bridging the gap between speech technology and natural language processing: an evaluation toolbox for term discovery systems", *Proc. Language Resources and Evaluation Conference*, 2014.

39. Sakoe, H. and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 26, No. 1, pp. 43–49, 1978.

40. Lin, Y.-L., T. Jiang and K.-M. Chao, "Efficient algorithms for locating the length-constrained heaviest segments with applications to biomolecular sequence analysis", *Journal of Computer and System Sciences*, Vol. 65, No. 3, pp. 570 – 586, 2002.

41. Johnson, J., M. Douze and H. Jégou, "Billion-scale similarity search with GPUs", *arXiv preprint arXiv:1702.08734*, 2017.

42. Newman, M. E. J., "Fast algorithm for detecting community structure in networks", *Physical Review E*, Vol. 69, No. 6, Jun 2004, `http://dx.doi.org/10.1103/PhysRevE.69.066133`.

43. Lyzinski, V., G. Sell and A. Jansen, "An evaluation of graph clustering methods for unsupervised term discovery", *Proc. Interspeech*, pp. 3209–3213, 2015.

44. Forster, J., C. Schmidt, T. Hoyoux, O. Koller, U. Zelle, J. Piater and H. Ney, "RWTH-PHOENIX-Weather: A Large Vocabulary Sign Language Recognition and Translation Corpus", *Language Resources and Evaluation*, pp. 3785–3789, Istanbul, Turkey, 2012.

45. Forster, J., C. Schmidt, O. Koller, M. Bellgardt and H. Ney, "Extensions of the Sign Language Recognition and Translation Corpus RWTH-PHOENIX-Weather", *Language Resources and Evaluation*, pp. 1911–1916, Reykjavik, Iceland, 2014.

46. Sutton, V., "Sign writing.", *Deaf Action Committee (DAC) for Sign Writing*, 2000.

47. Kristoffersen, J. H., T. Troelsgård, A. S. Hardell, B. Hardell, J. B. Niemelä, J. Sandholt, and M. Toft, *Ordbog over Dansk Tegnsprog*, 2008, `http://www.tegnsprog.dk`.

48. D. McKee, R. McKee, S. P. Alexander, and L. Pivac, *The Online Dictionary of New Zealand Sign Language*, 2015, `http://nzsl.vuw.ac.nz`.

49. Cao, Z., T. Simon, S.-E. Wei and Y. Sheikh, "Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields", *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1302–1310, 2017.

50. Karadayi, J. and M. Bernard, *Term Discovery Evaluation*, 2019, `https://github.com/bootphon/tdev2`, accessed in March 2020.

51. Jansen, A., B. Van Durme, and G. Sell, *ZRTools: Zero-Resource Speech Discovery, Search and Evaluation Toolkit*, 2015, `https://github.com/arenjansen/ZRTools`, accessed in May 2019.

APPENDIX A: DISCOVERED CLUSTERS

Table A.1. Largest clusters and most frequent labels using the SC based method, for the KNN (DH) setup, at 10% Coverage. The frequencies are in % of segments of the same label and are sorted by cluster sizes.

Clus. Size	**Avg. Len. (s)**	**Label 1**	**Freq 1**	**Label 2**	**Freq 2**	**Label 3**	**Freq 3**
11	0.7	MORGEN	63.6	WIE-AUSSEHEN;MORGEN	9.1	WIE-AUSSEHEN	9.1
6	0.6	KUEHL	33.3	WIND;WEHEN	16.7	STURM;__EMOTION__	16.7
3	0.3	KUEHL	66.7	FROST	33.3		
3	1.1	STURM	66.7	WEHEN	33.3		
3	0.5	KUEHL	33.3	KALT	33.3	FROST	33.3
2	0.8	loc-WEST;BLEIBEN	50.0	loc-NORD	50.0		
2	0.4	__EMOTION__;NAECHSTE	50.0	NAECHSTE	50.0		
2	0.9	SCHNEE	50.0	REGEN	50.0		
2	1.0	STURM	100.0				
2	0.3	UEBERWIEGEND	50.0	HAUPTSAECHLICH	50.0		
2	0.5	WIND	50.0	UNWETTER	50.0		
2	1.2	WIND	50.0	STURM	50.0		
2	0.8	KOENNEN;STURM	50.0	KOENNEN	50.0		
2	0.5	MONTAG	100.0				
2	0.5	STURM	100.0				
2	0.7	WIND	50.0	STURM	50.0		
2	0.8	UNWETTER	50.0	ORKAN	50.0		
2	0.5	KALT	50.0	FROST	50.0		

Table A.2. Largest clusters and most frequent labels using the OE based method, for the KNN (DH) setup, at 10% Coverage. The frequencies are in % of segments of the same label and are sorted by cluster sizes.

Clus. Size	**Avg. Len. (s)**	**Label 1**	**Freq 1**	**Label 2**	**Freq 2**	**Label 3**	**Freq 3**
46	0.6	KUEHL	19.6	FROST	15.2	STURM	13.0
36	0.4	MORGEN	97.2	MORGEN;MONTAG	2.8		
34	1.2	WIE-AUSSEHEN;MORGEN	50.0	WETTER;WIE-AUSSEHEN;MORGEN	38.2	MORGEN	5.9
25	0.8	WIND	32.0	STURM	16.0	KUEHL	16.0
9	0.5	MONTAG	66.7	MORGEN	22.2	MORGEN;MONTAG	11.1
8	0.6	MONTAG	87.5	MORGEN;MONTAG	12.5		
6	0.7	REGEN	100.0				
6	0.8	WIND	33.3	STURM	33.3	UNWETTER	16.7
5	0.3	MORGEN	100.0				
5	0.5	NACHT	80.0	HEUTE;ABEND	20.0		
5	1.1	WIND;SCHWACH;MAESSIG	20.0	WEHEN	20.0	MAESSIG	20.0
5	0.6	WIE-AUSSEHEN	80.0	WIE-AUSSEHEN;MORGEN	20.0		
5	0.5	NICHT-KALT	20.0	KUEHL;MEHR	20.0	KUEHL	20.0
5	0.5	AUSSEHEN;MORGEN	60.0	MORGEN	40.0		
4	0.5	WIE-AUSSEHEN	50.0	__ON__;WIE-AUSSEHEN	25.0	WETTER;WIE-AUSSEHEN	25.0
4	0.6	MONTAG	75.0	MORGEN	25.0		
4	0.7	NORD	100.0				
4	0.8	STURM	50.0	WIND	25.0	KOENNEN;UNWETTER	25.0
4	0.5	NACHT	50.0	REGEN	25.0	HEUTE;NACHT	25.0
4	1.1	WETTER;WIE-AUSSEHEN	50.0	__PU__;WETTER;WIE-AUSSEHEN	25.0	WETTER	25.0
4	0.5	MORGEN	50.0	MORGEN;MONTAG	25.0	MONTAG	25.0
4	0.5	FROST	100.0				
4	0.9	loc-NORDWEST	25.0	loc-NORD	25.0	NORDRAUM;NUR	25.0

Table A.3. Largest clusters and most frequent labels using the SC based method, for the KNN (OP) setup, at 10% Coverage. The frequencies are in % of segments of the same label and are sorted by cluster sizes.

Clus. Size	**Avg. Len. (s)**	**Label 1**	**Freq 1**	**Label 2**	**Freq 2**	**Label 3**	**Freq 3**
202	0.7	SCHNEE	19.8	WOLKE	14.4	REGEN	13.4
76	0.8	WETTER	15.8	MORGEN	9.2	FREUNDLICH	5.3
71	0.7	WETTER	31.0	SCHWACH	9.9	JETZT	7.0
13	0.5	KUEHL	46.2	poss-BEI-UNS;START	7.7	STARK	7.7
12	0.6	MORGEN	33.3	DIENSTAG	33.3	WIE-AUSSEHEN;MORGEN	8.3
10	0.5	TROCKEN	50.0	MILD	30.0	NASS	10.0
9	0.6	DESWEGEN	44.4	DAS-IST-ES	33.3	DESWEGEN;SCHAUER	11.1
7	0.7	MILD	28.6	SCHAUEN;REST;IN-DIESE-WOCHE;SCHNEE	14.3	FREUNDLICH	14.3
6	0.8	WETTER;MORGEN	66.7	WETTER;FUER;MORGEN	16.7	MORGEN	16.7
6	1.0	__PU__;FRISCH	16.7	WETTER	16.7	UNGEMUETLICH	16.7
6	1.0	SONNE	33.3	VIEL;SONNE	16.7	SONNE;REGION	16.7
6	1.2	VORBEI;REGEN	16.7	TROCKEN;VIEL	16.7	TEIL;FREUNDLICH;WOLKE	16.7
6	0.6	DARUM	50.0	DESWEGEN	33.3	BLEIBEN;__EMOTION__;HOCH	16.7
5	1.0	FREUNDLICH	80.0	TEIL;FREUNDLICH	20.0		
5	1.1	WIE-AUSSEHEN;MORGEN	40.0	WETTER;WIE-AUSSEHEN;MORGEN	20.0	WETTER;WIE-AUSSEHEN	20.0
4	0.6	IN-KOMMEND	100.0				
4	1.3	STARK;SCHNEE;HEUTE	25.0	ODER;SCHNEE	25.0	MEHR;FREUNDLICH;ACHTZEHN	25.0

Table A.4. Largest clusters and most frequent labels using the OE based method, for the KNN (OP) setup, at 10% Coverage. The frequencies are in % of segments of the same label and are sorted by cluster sizes.

Clus. Size	**Avg. Len. (s)**	**Label 1**	**Freq 1**	**Label 2**	**Freq 2**	**Label 3**	**Freq 3**
127	0.6	WOLKE	18.1	SCHNEE	17.3	REGEN	15.0
121	0.7	WETTER	22.3	FREUNDLICH	11.6	SCHWACH	9.1
56	0.8	MORGEN	17.9	MONTAG	7.1	TAG	5.4
48	0.8	SCHNEE	37.5	REGEN	6.2	NACHT	4.2
46	0.7	DESWEGEN	17.4	REGEN	10.9	SCHAUER	6.5
29	0.9	NORD	13.8	KLAR	6.9	FROST	6.9
26	1.0	ZWANZIG	7.7	REGEN;GEWITTER	7.7	REGEN-PLUSPLUS;GEWITTER	7.7
23	0.9	WIE-AUSSEHEN	21.7	WETTER	21.7	WIE-AUSSEHEN;MORGEN	13.0
23	0.9	WIE-AUSSEHEN;MORGEN	13.0	MORGEN	13.0	ZWANZIG	8.7
21	0.7	WIE-AUSSEHEN;MORGEN	14.3	MORGEN	14.3	DIENSTAG	14.3
20	0.6	TROCKEN	30.0	MILD	20.0	LIEB;ZUSCHAUER	10.0
19	0.6	KALT	21.1	KUEHL	15.8	GEFRIEREN	10.5
19	0.8	LEICHT;MAESSIG	10.5	loc-REGION-PLUSPLUS	5.3	ZUSAMMENHANG;SCHWACH;MAESSIG	5.3
18	1.1	REGEN	16.7	KOENNEN	11.1	GEWITTER;KOENNEN	11.1
16	0.6	WIE-AUSSEHEN	25.0	SECHSZEHN	25.0	DIENSTAG	12.5
13	0.9	NORD	23.1	WIND	15.4	loc-NORD	7.7
11	0.7	WOLKE	27.3	REGEN	27.3	cl-KOMMEN	18.2
11	0.9	SCHNEE	27.3	WOLKE;AUFLOESEN;ABER;DEUTSCH	9.1	WECHSELHAFT;KOENNEN	9.1
10	0.5	IN-KOMMEND	90.0	NAEHE	10.0		
10	1.0	NACHT	40.0	WIND	10.0	NACHT;ZWEI;BIS	10.0
9	1.0	loc-SUEDOST	11.1	__EMOTION__;REGION;KUESTE	11.1	SUEDRAUM;BISSCHEN	11.1

Table A.5. Largest clusters and most frequent labels using the SC based method, for the E-SDTW (DH) setup, at 10% Coverage.

The frequencies are in % of segments of the same label and are sorted by cluster sizes.

Clus. Size	Avg. Len. (s)	Label 1	Freq 1	Label 2	Freq 2	Label 3	Freq 3
107	2.0	BISSCHEN;WOLKE	1.9	WOLKE;NEBEL	1.9	REGION;SCHNEE	1.9
68	2.3	WETTER;WIE-AUSSEHEN;MORGEN;MONTAG	13.2	JETZT;WETTER;WIE-AUSSEHEN;MORGEN	8.8	WETTER;WIE-AUSSEHEN;MORGEN	4.4
45	2.2	__ON__;JETZT;WETTER;WIE-AUSSEHEN;MORGEN	13.3	__ON__;JETZT;WETTER;WIE;AUSSEHEN;MORGEN	8.9	JETZT;WETTER;WIE;AUSSEHEN;MORGEN	6.7
37	2.1	KOMMEN;KALT	5.4	loc-SUED;FROST	2.7	cl-KOMMEN;STARK	2.7
33	2.0	loc-WEST;BLEIBEN	3.0	NACHT;KUEHL;KLAR	3.0	FEUCHT;KALT;NORD	3.0
32	2.0	DIENSTAG	6.2	DIENSTAG;MEHR;WARM	6.2	__ON__;DIENSTAG;__EMOTION__	3.1
24	1.9	cl-KOMMEN	12.5	SONNE	8.3	__EMOTION__;cl-KOMMEN;WOLKE;cl-KOMMEN;DANN	4.2
10	2.0	loc-NORDWEST	10.0	loc-NORDOST	10.0	loc-NORD;__EMOTION__	10.0
8	2.2	__ON__;WETTER;WIE-AUSSEHEN;MORGEN;SONNTAG	12.5	__ON__;WETTER;WIE-AUSSEHEN;MORGEN;SAMSTAG	12.5	__ON__;WETTER;WIE-AUSSEHEN;MORGEN	12.5
6	2.2	__ON__;IX;TIEF	16.7	__EMOTION__;NAECHSTE;WOCHE;__EMOTION__	16.7	TIEF;loc-KOMMEN;NAECHSTE;WOCHE	16.7
6	2.1	TRUEB;BISSCHEN;REGEN	16.7	REGEN;MITTE;SUED	16.7	MITTE;SUED	16.7
5	1.9	__ON__;ZWEIFEL;SO	20.0	WEHEN-PLUSPLUS	20.0	WEHEN	20.0
5	2.1	loc-NORD;BLEIBEN;KUEHL;VIEL	20.0	SUED;SCHWACH;WEHEN	20.0	NORD;BLEIBEN;KUEHL;MEISTENS	20.0
4	2.1	SCHNEE;__EMOTION__;REGEN;GLATT	25.0	SCHNEE;UND;REGEN;GLATT	25.0	SCHNEE;UND;REGEN;EIS;UND	25.0
4	1.9	cl-KOMMEN;__LEFTHAND__	25.0	NICHT-MEHR;ABER	25.0	NACHMITTAG;__EMOTION__	25.0
4	2.3	WOLKE;TEIL;SCHAUER	25.0	WOLKE;TEIL;REGEN	25.0	NACHT;WOLKE;TEIL;__EMOTION__;REGEN	25.0
4	2.3	VIER;TAG;SECHSZEHN	25.0	TAG;loc-BERG;VIER	25.0	TAG;DREI;ALPEN	25.0
4	2.0	__LEFTHAND__	25.0	__EMOTION__;WEST;loc-REGION;GEWITTER	25.0	UMWANDELN;SCHNEE	25.0
3	2.1	SCHNEE	33.3	MAL;LEICHT;SCHNEE;SPEZIELL	33.3	LEICHT;REGEN	33.3
3	1.9	__ON__;IN-KOMMEND;loc-NORD	33.3	__ON__;IN-KOMMEND	33.3	GLEICH;IN-KOMMEND	33.3
3	2.1	NACHT;KOENNEN;__EMOTION__;AUTO	33.3	NACHT;KOENNEN;FROST;__OFF__	33.3	HEUTE;NACHT;KUEHL	33.3
3	2.1	KLAR;HIMMEL;STERN;KOENNEN;SEHEN	33.3	HIMMEL;STERN;__EMOTION__;SEHEN;DESHALB	33.3	HIMMEL;KLAR;HIMMEL;STERN;KOENNEN;SEHEN	33.3
3	2.3	loc-NORD;MINUS;DREI;FLUSS	33.3	loc-NORD;BIS;DREISSIG	33.3	NORD;ZWEIDREISSIG	33.3
3	2.0	MORGEN;EINFLUSS;WETTER	33.3	MONTAG;FRAGEZEICHEN;NOCH-NICHT	33.3	MINUS;SECHS;IN-PAAR-TAGE-SPAETER	33.3

Table A.6. Largest clusters and most frequent labels using the OE based method, for the E-SDTW (DH) setup, at 10% Coverage. The frequencies are in % of segments of the same label and are sorted by cluster sizes.

Clus. Size	Avg. Len. (s)	Label 1	Freq 1	Label 2	Freq 2	Label 3	Freq 3
253	2.1	__ON__;JETZT;WETTER;WIE-AUSSEHEN;MORGEN	5.1	__ON__;JETZT;WETTER;WIE;AUSSEHEN;MORGEN	2.8	__ON__;JETZT;WETTER;MORGEN	2.4
17	1.9	__ON__;KALT;VON	5.9	IX;WIND;BERG	5.9	DUNST;TEIL;FROST	5.9
11	1.9	loc-WEST;BLEIBEN	9.1	loc-REGION;NIEDERSACHSEN	9.1	loc-OST;loc-WIND	9.1
9	1.9	MORGEN;ABEND;KOMMEN	22.2	cl-KOMMEN;DICK;cl-KOMMEN;__EMOTION__	11.1	__EMOTION__;WOLKE;cl-KOMMEN	11.1
6	2.0	loc-OST;loc-NORD;KOMMEN	16.7	loc-NORDWEST	16.7	loc-NORD;__EMOTION__	16.7
5	2.0	WIND;DURCH	20.0	WIND;ABER	20.0	KUEHL;DARUM	20.0
5	2.2	cl-KOMMEN;IX;MORGEN;KOMMEN	20.0	__ON__;IX;TIEF;KOMMEN	20.0	TIEF;loc-KOMMEN;NAECHSTE	20.0
5	2.1	KUEHL;NORD;REGEN	20.0	KOENNEN;GEFRIEREN	20.0	FROST;loc-NORD	20.0
5	2.1	LUFT;MILD	40.0	WIND;VERSCHWINDEN;__EMOTION__;BLEIBEN	20.0	KUEHL;UND;NASS;ABER	20.0
5	2.1	cl-KOMMEN	20.0	REGEN;AUFZIEHEN	20.0	GROSSBRITANNIEN	20.0
5	1.9	FREUNDLICH;DIENSTAG;SONNE	20.0	DIENSTAG;cl-KOMMEN	20.0	DIENSTAG;ZUERST	20.0
4	2.1	STURM;MOEGLICH	25.0	MOEGLICH;__PU__	25.0	KUESTE;KOENNEN;STURM;__EMOTION__;REGEN	25.0
4	2.1	SIEBEN;ZWANZIG;FEBRUAR;ZEIGEN-BILDSCHIRM;__OFF__	25.0	SIEBEN;ZWANZIG;FEBRUAR;ZEIGEN-BILDSCHIRM	25.0	MONTAG;ACHT;ZWANZIG;FEBRUAR;ZEIGEN-BILDSCHIRM;...	25.0
4	2.1	loc-NORDOST;SCHNEE	25.0	loc-NORD;__EMOTION__;REGEN-PLUSPLUS	25.0	NORDRAUM;REGEN;SCHNEE	25.0
4	2.1	cl-KOMMEN	25.0	__EMOTION__;__LEFTHAND__;cl-KOMMEN;__LEFTHAND_...	25.0	SONNE;__LEFTHAND__;WOLKE	25.0
4	2.0	loc-NORDWEST;WOLKE	25.0	loc-NORDWEST;KOMMEN;SCHAUER-PLUSPLUS	25.0	NORDWESTRAUM;cl-KOMMEN	25.0
3	2.2	STURM;MOEGLICH;NORD	33.3	NACHT;KUEHL;KLAR	33.3	AUCH;STURM;MOEGLICH;SOLL	33.3
3	2.0	loc-WEST;BLEIBEN	33.3	loc-OST;loc-WIND	33.3	loc-NORDWEST;NOCH	33.3
3	2.0	__ON__;MONTAG;UEBERALL;SONNE	33.3	MONTAG;MEISTENS;SONNE	33.3	MONTAG;MEHR;SONNE;WOLKE	33.3
3	2.0	__ON__;ZWEIFEL;SO	33.3	DESWEGEN;DAZWISCHEN	33.3	DARUM;ICH;KOMMEN;KUEHL;KOMMEN-PLUSPLUS	33.3
3	1.9	KUESTE;NULL	33.3	IX;TIEF	33.3	IX;KOMMEN	33.3
3	1.9	FREUNDLICH	33.3	BLEIBEN;__EMOTION__;FREUNDLICH	33.3	BLEIBEN;TROCKEN;EINIGE	33.3
3	2.2	KOENNEN;HIMMEL;STERN;KOENNEN;SEHEN	33.3	HIMMEL;STERN;__EMOTION__;SEHEN;DESHALB	33.3	HIMMEL;STERN;KOENNEN;SEHEN;ABER	33.3
3	2.3	__ON__;WETTER;WIE-AUSSEHEN;MORGEN;SONNTAG	33.3	__ON__;JETZT;WETTER;WIE-AUSSEHEN;MORGEN;SONNTAG	33.3	__ON__;DANN;WETTER;WIE-AUSSEHEN;MORGEN;MITTWOCH	33.3