arXiv:1308.3839v1 [cs.CL] 18 Aug 2013

Consensus Sequence Segmentation

Tamal Chowdhury, Rabindra Rakshit, and Arko Banerjee

College of Engineering and Management, Kolaghat, WB, India
tamalasp@gmail.com, rovir2r@gmail.com, arko.banerjee@gmail.com

Abstract. In this paper we introduce a method to detect words or phrases in a given sequence of alphabets without knowing the lexicon. Our linear time unsupervised algorithm relies entirely on statistical relationships among alphabets in the input sequence to detect location of word boundaries. We compare our algorithm to previous approaches from unsupervised sequence segmentation literature and provide superior segmentation over number of benchmarks.

Keywords: text segmentation, sequence segmentation, unsupervised segmentation, entropy.

1 Introduction

Finding interesting patterns within a data that is expressed as a long sequence of discrete symbols becomes useful in applications such as extracting new technical terms, indexing documents for information retrieval, and correcting optical character recognition. In such cases segmenting sequences automatically into meaningful blocks like phrases or words, becomes beneficial. Work related to text segmentation that have been reported in literature are mostly supervised in nature. Use of supervised segmentation methods like lexico-syntactic morphological analyzers are not robust across different domains as terms from a certain domain may not be in the lexicon ,unless efforts are made to modify their databases accordingly.As an alternative to supervised approaches, a limited amount of work on unsupervised text segmentation have been reported in literature. Sequitur[1] is an unsupervised, compression-based algorithm that builds a context-free grammar from a sequence of discrete tokens. Sequitur is designed to learn the hierarchical structure of sequences that can be used for segmenting sequences [2]. The Voting Experts (VE) by Cohen et al[2] shows that the log frequency of a segment is a measure of its internal entropy. The VE exploits the signature of segments to find word boundaries in text from four languages and episode boundaries in the activities of a mobile robot. In the paper A Statistical Learning Algorithm for Word Segmentation by Jerry Van Aken[4] a Viterbi trellis is used to simultaneously evaluate a set of hypothetical segmentations of a block of adjacent words.The paper considers statistical assumptions that may not be feasable in general cases. Brent [5] presents a model-based, unsupervised algorithm for recovering word boundaries in a natural-language text from which

they have been deleted. The algorithm is derived from a probability model of the source that generated the text but does not estimate a probability distribution on words; instead, it attempts to calculate the prior probabilities of various word sequences that could underlie the observed text. Statistical learning by 8-month-old infants by Saffran et al [6] shows that segmentation of words from fluent speech can be accomplished by eight-month-old infants based solely on the statistical relationships between neighboring speech sounds.

In this paper we propose a simple and efficient unsupervised sequence segmentation method which learns mostly from frequency of occurrences of alphabets , thus avoiding the costs of building lexical or syntactic knowledge. The key advantage of our method is that it gives good performance even if the input data is very small in size. The segmentation model we introduce in this paper involves three main phases: Sequence Memory design, probabilistic sequence segmentations and finally consensus of derived sequence segmentations. The input sequence is prepared by converting documents into ordered sequence of words by removing spaces between them. To detect recurring substrings in the input, a *sequence memory* storage is designed to efficiently store all possible substrings along with their frequency of occurrences. By extracting frequency information from the *sequence memory* the sequence segmentation process generates a set of feasible segmentations of the input sequence on basis of different probabilistic measures. Finally, a consensus of the derived segmentations is performed to achieve a more accurate and robust segmented sequence. We compare the performance of our method to VE and another improved version of VE called Bootstrap Voting Algorithm (BVE) by Cohen et al [3], over standard benchmarks from the literature and show that in both cases our method outperform both the algorithms.

The rest of this paper is organized as follows. Section 2 describes the VE and BVE algorithms and their drawbacks. Section 3 describes the Sequence Memory design. Section 4 and 5 describe entropy and frequency as segmentation measures, respectively. Sequence segmentation procedure is discussed in section 6. Idea of consensus has been introduced in section 7. In section 8 we compare our method with VE using a toy example. Section 9 shows empirical results of our method on standard benchmakrs. Finally section 10 concludes the paper.

2 Voting Experts Algorithm and its drawbacks

Cohen et al [2] in their Voting Experts algorithm uses two experts one that minimizes the internal entropy of segments, and one that maximizes the frequency of segments to detect segment boundaries. The algorithm moves a sliding window of a fixed size over the sequence from left to right and at each position of the window each expert votes for the most probable boundary. The final boundary is introduced where the sum of votes at each position exceeds a threshold value provided by the user. Voting Experts claimed to outperform Sequitur[1] on word boundary detection from different languages, and in identifying robot motion sequences.

To detect probable boundaries in the window VE sweeps a pointer inside from left to right. VE claims that summation of frequencies of two substrings on both sides of the pointer contained by the window should give a good measure for detecting a boundary. For example in Figure 1 WORD1 and WORD2 are two words in the sequence and let us assume at an iteration the window contains ORD1WOR. When the pointer points the position of the solid line, summa-

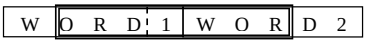

Fig. 1. Detection of boundary inside window

tion of frequencies of ORD1 and WOR is higher than that of other positions. For example, if the pointer points the position of the dashed line then it is a high chance that the frequency of 1WOR is low, since WORD2 has less chance of occurring just after WORD1. Hence summation of frequencies of ORD and 1WOR becomes low. But the problem arises when the pointer splits two substrings where one of them contains one or two alpahbets. One alphabet or two consecutive alphabets in a sequence can occur very frequently, hence the sum of frequencies of the two substrings dominates over the most probable boundary position. The problem can be partially tackled by normalizing the frequency measure for different length of substrings. The second drawback of the method is the fixed size of window length. Considering the window size large makes the window contain more than two words in most iterations, hence giving less chance of separating words properly. Again a small window size is unabled to evaluate actual frequency of most of the words. Therefore ideal case should be when the window contains exactly two words, which may not happen frequently in the sequence. That is why a fixed window size would yield unnecessary segments. VE implements boundary entropy as another expert for detecting word boundaries. Since VE applies boundary entropy from left to right it is only able to detect end boundaries of words but unable to detect the start boundaries. Though end boundary of a word is the start bounday of the next word but sometimes due to failure in detecting boundary by entropy expert, VE offers no option to correct the missed boundary by reversely (right to left) implementing the entropy expert. Conceptually and experimentally boundary entorpy expert shows better result on detecting boundary with respect to frequency expert. To improve accuracy in boundary detection, VE considers summation of frequency and entropy votes at each position of the window. Though sum of votes may improve the robustness of the resulting segmentation but it may also generate extra unnecessary boundaries, resulting in oversegmented sequence . Last but not the least is the selection of optimum threshold by the user, which along with fixed window length restrict VE to be fully automatic.

Cohen et al proposed Bootstrap Voting Experts (BVE) [3], an extension to the VE algorithm for unsupervised sequence segmentation. BVE generates a

series of segmentations, each of which incorporates knowledge gained from the previous segmentation by maintaining a *knowledge tree*. The paper claims that the method of bootstrapping improves the performance of Voting Experts in a variety of unsupervised word segmentation scenario. But by maintaining the same structure of VE, BVE is unable to escape from the said drawbacks of VE. Though for considering knowledge tree as another voting expert it is able to remove unnecessary boundaries by providing proper user given thresholds, but due to maintainance of the knowledge tree it suffers from higher space and time complexity for large benchmark datasets.

3 Sequence Memory

A successful sequence segmentation algorithm seeks to maximize the unpredictability between substrings or segments. To do that, alphabet that succeeds and alphabet that precedes a segment, should have their unpredictability maximized with respect to the segment. The boundary entropy of a segment is a measure that expresses the magnitude of unpredictability of its end boundary by traversing the sequence from left to right. Due to unsupervised nature of our procedure finding start boundary of a segment can also be determined using boundary entropy by traversing the sequence from right to left. Hence we propose a bidirectional measurement of boundary entropy of each segment in the sequence to measure unpredictablility of its both start and end boundaries. In following sections we will call this as forward/backward boundary entropy measurement. We design the Sequence Memory by introducing a bidirectional TRIE data structure to evaluate the bidirectional boundary entropy efficiently.

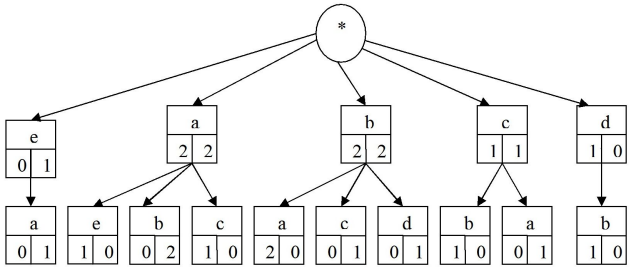

Fig. 2. Bidirectional TRIE structure of the sequence {e a b c a b d}

To represent all the segments and reverse segments of length n in a sequence, an ngram TRIE of depth $n + 1$ is generated by sliding a window of length n forward and backward across the sequence. The window length is assumed to be at least the length of the largest word plus one. Each node of the TRIE consists of two frequency fields to represent a segment and the corresponding

reverse segment. For example, the sequence $\{e\ a\ b\ c\ a\ b\ d\}$ generates the TRIE in Figure 2 with depth 3. Every segment and the reverse segment of length 2 or less in the sequence is represented by a node in the tree. The right and left frequency fields represent frequency of the segment and the reverse segment, respectively. For example, the segment $\{a\ b\}$ and the reverse segment $\{b\ a\}$ occurs twice. The reverse segment $\{a\ c\}$ occurs once but the segment $\{a\ c\}$ does not occur. We denote the frequencies of a segment and a reverse segment of a node n_i by f_i^1 and f_i^2, respectively. The m number of children of n_i, denoted by $n_{i,1}, ..., n_{i,m}$, have frequencies $f_{i,1}^t, ..., f_{i,m}^t$, respectively, where $t = 1, 2$. In the next section we will use the notations to formulate boundary entropy of a segment.

4 Boundary Entropy as segmentation measure

The conditional probability of an alphabet occurring after/before a segment is measured by the frequency of the alphabet occurring after/before the segment divided by the frequency of the segment, which we denote by $Pr(n_{ij}^t) = f_{ij}^t / f_i^t, t = 1, 2$. The boundary entropy of the segment is the summation of entropy of the conditional probabilities of all such alphabets. Then the boundary entropy of the segment and the reverse segment can be calculated as $H(n_i) = -\sum_{j=1}^{m} Pr(n_{ij}^t) \log Pr(n_{ij}^t), t = 1, 2$.

For example, the node **a** as a segment in Figure 1 has low entropy equal to 0 because it has only one child **b**. Hence there is a chance that $\{a\ b\}$ forms a segment. Whereas $\{b\}$ as a segment with high entropy equal to 1.0 has a high chance of being the end of a segment. Therefore we expect the segmented sequence as $\{\ e\ a\ b^*\ c\ a\ b^*\ d\ \}$, where * denotes the end of a segment. Again $\{b\}$ as a reverse segment has an entropy equal to 0 hence high chance of$\{b\ a\}$ to form a reverse segment. Whereas $\{a\}$ as a reverse segment has an entropy equal to 1.0 hence a high chance of being the end of a reverse segment. Therefore after reverse segmenting the expected segmented sequence should be $\{e\ \#a\ b^*c\#a\ b^*d\}$, where # denotes the end of a reverse segment. We have implemented the bidirectional TRIE structure to extract segments out of sequences and got pretty accurate results. In the following section we utilize the frequency information of substrings in the sequence to perform segmentation.

5 frequency as segmentation measure

The motivation in this section is to determine the boundary between two words by comparing frequencies of the words with substrings that straddle the common boundary of the words. We explain the situation using the following example. Figure 3 shows a sequence of ten alphabets W O R D 1 W O R D 2 that represents two consecutive words WORD1 and WORD2. The goal is to determine whether there should be a word boundary between "1" and "W". Let us assume that at an instance a window of length 10 contains the words and it considers a hypothetical boundary in the middle that is between WORD1 and WORD2. To check whether the boundary is a potential one we count the number of times the

two segments (words WORD1 and WORD2) that are adjacent to the boundary are more frequent than other segments inside the window of same length that straddle the boundary. Figure 3 shows four possible straddling words of length five. For example, frequency of one of the straddling words 1WOR should be low compared to WORD1 or WORD2, since WORD2 has less chance of occurring just after WORD1. We maintain a score for the hypothetical boundary to measure its potentiality and it is incremented by one only if WORD1 or WORD2 are more frequent than that of 1WOR. So, if both possible words are higher in frequency than a straddling word the score to the hypothetical boundary location is incremented twice. In our algorithm we consider a window of length $2n$, where n varies from 2 to a maximum window length. A window of length $2n$ has its hypothetical boundary separating alphabets at n th and $n+1$ th positions, hence number of straddling words across the boundary would be $n-1$. Therefore, by sliding the internal window along the sequence each position (except end positions) gets total $2(n-1)$ scores. To normalize the scores of different window length we average the scores by dividing it with $2(n-1)$.

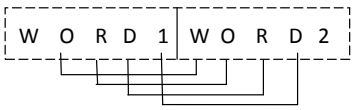

Fig. 3. Detection of word boundary by comparing frequencies of WORD1 and WORD2 with all four possible straddling semgments of length five

There are several advantages of using this frequency model over VE. Since comparison of segments of same length are done here, so there is no necessity of frequency normalization. Unlike VE due to varying size, the window can contain any two consecutive words of different length for some values of n, where either of the words can contribute properly to the boundary score. For a very long window the left and right windows would contain more than one word. In that case the segments in the left and right windows would occur only once in the sequence and also it is also same for all straddling segments accross the boundary. So they would contribute almost nothing to the boundary score. As window length is increased, after a certain length the contribution becomes mostly zero. Therefore the maximum window length parameter is taken at least twice the length of the largest word. The only input paramenter of our algorithm that is the window size does not affect much on the quality of output for small variations. In the next section we explain the procedure of segmentation when we have already gathered boundary information of a sequence using entropy and word frequencies.

6 Segmentation Procedure

We implemented three separation measures namely, forward and backward boundary entropies and frequency, to perform segmentation of sequences. For each iteration i.e. for each shift of windows every position in the sequence inside the window receives a separation measure for finding segments. Therefore each position receives total n number of separation values for segmentation, where n is the length of the window. Instead of taking threshold values from the user the algorithm cuts the text at locations that have locally maximum separation values. In later sections we will use abbreviation of sequence segmenation by frequencey as SSF, sequence segmentation by forward entropy as SSFE and sequence segmentation by backward entropy as SSBE.

7 Consensus

For more accurate and robust segmentation we make consensus of three segmented sequences derived by SSF, SSFE and SSBE into a final sequence. Cuts that are common in all three segmentations are considered to be very accurate but less in number. To get moderate number of accurate cuts we consider cuts that are common in at least two segmentations.

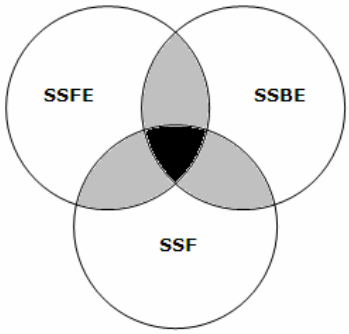

Fig. 4. Venn diagram representation of SSFE, SSBE and SSF. The shaded part represents CS, where in the darker shade all the three methods agree

The output segmentation is considered as the final consensus segmentation, which we will abbreviate as CS. In the following section we compare our method with Voting Experts algorithm using a Toy example.

8 Comparison with VE

We have chosen some sentences from the paper of Cohen et al and ran VE, BVE and our proposed method to compare their results. We separately evaluate

the three segmentations- SSF, SSFE, SSBE. Finally we show the result of the consensus (CS) of the three segmentations .An optimum threshold is chosen for VE for the following input.

Input: Miller was close to the mark when he compared bits with segments. But when he compared segments with pages he was not close to the mark. his being close to the mark means that he is very close to the goal. segments may be identified by an information theoretic signatures. page may be identified by storage properties. bit may be identified by its image.

Output of VE: Mi*lle*rwas*close*tot*he*ma*rkwhe*nhe*co*mpa*re*dbi*ts wi * thse*gm*ents*But*whe*nhe* co*mpa*re*dse*gm*ents*wi*thpa*ge*she*was*not* close * tot*he*ma*rkhi*sbe*ingc*lose*tot*he*ma*rkme*ans*that*he*isve*ryclose *tot*he*goa*lse*gm*ents*ma*ybe*ide*nti*fie*dbya*ni*nfo*rmati*ont*he*ore*ti* csi *gnat*ure*spa*ge*ma*ybe*ide*nti*fie*dbys*tora*ge*pro*pe*rtie*sbi*tma*y be*ide *nti*fie*dbyi*tsi*mage

Output of BVE: Mi*lle*rwas*close*to*the*ma*rkwhe*nhe*co*mpa*re*db*it*swi *th*segm*en*ts* But*whe*nhe*co*mpa*re*dse*gm*en*ts*wi*th*pa*ge*she*was*not *close*to*the*ma*rk*his* bei*ngc*lose*to*the*ma*rkme*ans*that*he*isve*ry close*to*the*go*al*segm*en*ts*ma* ybei*den*ti*fie*dby*an*inf*or*ma*ti*on*the *ore*ti*csi*gn*at*ure*spa*ge*ma*ybei*den *ti*fie*dby*st*or*age*pr*op*er*ti* esb*it*ma*ybei*den*ti*fie*dby*its*im*ag*e

Output of SSF: Mill*er*was*clo*se*to*the*mark*when*he*comp*ar*ed *bitswith *seg* ments*But*when*he*comp*ar*ed*seg* ments* with*page*she*was*not *clo*se *to* the*mark*his*bei*ng*clo*se* to*the *mark* me*an*stha*the*isvery *clo*se*to* the*goal*seg*ments* may*beid* ent*ifi*edby*an*inf*or*ma*tion *the*oretic*si*gn*atu* res*page*ma*ybeid*ent*ifi*edby* stor*age*prop*er*ties*bit *may*beid*ent*ifi*edby*itsi*mage

Output of SSFE: Miller*was*close*tothe*ma*rk*whe*nhe*compa*red* bits*with* se*gments*Butwhe*nhe*compa*red* se*gments* with*pa*ge*she*was* not*close*tothe*ma*rk*hisb*ei*ngclose*tothe*ma*rk* me*an*sth*at*he*isver*yclose*tothe* goalse*gments*ma* ybei*dent *ifie*dby*an*in*for*ma*ti*ont*he*or*eti*csi*gnat*ures*pa* ge*ma*ybei*dent*ifie*dby*st*or*age* proper*ti*es*bit*ma*ybei*dent*ifie*dby*its*ima*ge

Output of SSBE: Mill#er#was#clo#seto#the#mark#when#hecom#par#edbi#tswi #th#segm#en #tsBut#when#hecom#pa# red#segm#en#tswi#th#pag#es# he#wasn#ot#clo#seto#the#markhis#be#ing#clo#seto#the#mark#me#ans#tha #the#isv#ery#clo#seto# thegoal#segm#en#ts#may#beid#en#tifi#edby# an#inf#or#ma#tion#the#ore# tic#sign#atur#es#page#may#beid# en#tifi#edby#st# or#ageprop#er#ti#es#bit#may#beid#en#tifi#edby#it#sim#age

Output of CS: Mill*er*was*clo*se*to*the*mark*when*he*compar*ed*

bitswith* segments*But*when*he*compa*red* segments*with*page*she*was* not* clo*se* to*the*mark*his* bei*ng*clo*se *to*the*mark*me*an*stha*the* isvery *clo*se*to *the*goal* segments*may*beid*ent*ifi*edby*an*inf*or*ma*tion* the*oretic*si*gn*atures*page*ma*ybeid* ent*ifi*edby*st*or*age*prop*er*ti*es* bit*may*beid*ent*ifi*edby*itsimage

We see that result of BVE does not improve much from VE. CS forms a more robust segmenation by combining SSF, SSFE and SSBF. CS clearly performs better than that of VE as well as BVE.

9 Experiments

In this section, we present an empirical evaluation of our segmentation method in comparison with VE on a number of benchmark data sets[8][7].

Table 1. Detailed description of datasets along with comparison of VE and CS

Data	Source	points	words	classes	F_1 score of VE	F_1 score of CS
Tr31	TREC	927	10128	7	0.58	0.71
Tr41	TREC	878	7454	10	0.55	0.685
Tr45	TREC	690	8261	10	0.56	0.7
re0	Reuters-21578	1504	2886	13	0.52	0.668

The results of only four document datasets are recorded as for most of other datasets we got almost similar results. The first five columns of Table 1 summarize the basic properties of the data sets. To compare the performances of CS with VE, F_1 score[9] is used to determine quality of segmentation from both methods. The F_1 score is the harmonic mean of precision and recall and reaches its best value at 1 and worst score at 0. In the 6^{th} and 7^{th} column of Table 1 F_1 scores of VE and CS are recorded, respectively. The results show that CS achieves better performance on all the four data sets.

10 Conclusions and Future Work

In this paper, we proposed a new approach to text segmentation which is based on a probabilistic model that uses the idea of bidirectional sequence segmentation. In the future, we want to do a deeper analysis on the underlying reason for good performance of our algorithm and also to understand when it fails to give good results. We are also looking forward to compare our method with other unsupervised segmentation algorithms. We need to develop more sound concept that would resist in generating unnecessary small segments. We will also try to device better consensus technique to remove noise and achieve better segmenation.

References

1. C. Nevill-Manning, , and I. Witten. Identifying hierarchical structure in sequences: A linear-time algorithm. Journal of Artificial Intelligence Research (1997) 7:6782
2. Cohen, P., Adams, N., Heeringa, B.: Voting experts: An unsupervised algorithm for segmenting sequences. Journal of Intelligent Data Analysis. (2006)
3. Hewlett, D., Cohen, P.: Bootstrap Voting Experts. IJCAI (2009) 1071–1076
4. arXiv:1105.6162
5. Brent, M., An Efficient, Probabilistically Sound Algorithm for Segmentation and Word Discovery, Machine Learning, 34, 71-106.
6. Saffran, Jenny, Aslin, R., Newport, E.: Statistical Learning by 8-Month-Old Infants. Science, 274, (December1996) 1926-1928
7. Lewis, D.D.: Reuters-21578 text categorization test collection, http://www.daviddlewis.com/resources/testcollections/reuters21578
8. TREC: Text REtrieval Conference, http://trec.nist.gov
9. Van Rijsbergen, C.J.: Information Retrieval. 2nd edition. Dept. of Computer Science, University of Glasgow (1979)