On Homogeneous Segments

Robert Batůšek, Ivan Kopeček, and Antonín Kučera

Faculty of Informatics, Masaryk University
Botanicka 68a, 602 00 Brno
Czech Republic
{xbatusek,kopecek,tony}@fi.muni.cz

Abstract. Properties of homogeneous segment sets are investigated in this paper. A universal characterization of homogeneous sets is presented in the form of a polynomial algorithm which decides whether or not a set S is homogeneous with respect to S^*. It is shown that any set of homogeneous segments can be reduced to the uniquely determined minimal base and a polynomial algorithm solving this task is presented as well. Further, an efficient algorithm for finding the segment representation of a word by homogeneous segments is provided.

1 Introduction

Investigating segment set properties is related in particular to the applications in concatenative speech synthesis (see e.g. [1, 3, 6, 8, 10]) where we are trying to get optimal segment databases. The optimality usually involves contradicting requirements to the segment set; on one hand, we need the segment set to be as small as possible, on the other hand it should maximally respect coarticulation, which leads mostly to longer segments and large databases. To master these complex problems, an understanding of the properties and the structure of segment sets is of considerable value.

Basic taxonomy of the segments is given in [8]. Some other related work can be found in [2, 3]. In this paper, we investigate homogeneous segments. We show that this type of segments has particular properties, that are of theoretical interest and can be exploited in applications.

In the text we use standard terms and notation of the theory of formal languages and automata. If M is an alphabet (i.e. a finite nonempty set), then M^* will denote the free monoid over the set M, i.e. the set of all strings consisting of the elements of the set M (including the empty string). $card(M)$ denotes the cardinality of M, i.e. (for finite sets) the number of elements belonging to the set M.

2 Basic Types of Segments

In this section we briefly recall the definition of homogeneous segments and some other basic types of segment sets (see [3, 8]). Let us first briefly mention the notation and present basic terminology. Let A be an alphabet and S a finite nonempty subset of A^* not including the empty word. The set S is interpreted as a segment set (segment database). S^* denotes the set of all strings that can be obtained by concatenating the elements of S.

Further, let C be a language over the alphabet A, i.e. a nonempty subset of A^*. C will be interpreted as a corpus. Here, we are slightly generalizing the concept presented in [8], where the corpora are considered to be finite. In what follows, this generalization appears to be convenient.

In concatenative speech synthesis problems, this abstraction may be interpreted, for example, by considering A to be the set of phonemes, S to be the set of syllable segments and C to be a speech corpus. The following definitions present a basic classification of speech segments [8] in a slightly generalized form.

Definition 1. *A set of segments $S = \{s_1, s_2, \ldots, s_n\}$ is C-compatible (compatible with C), if for any $u \in C$ there are $s_i, s_j, \ldots, s_k \in S$ such that $u = s_i s_j \ldots s_k$. We denote $S(C)$ the set of all sets of segments compatible with C.*

Definition 2. *A C-compatible set S is consistent, if each element of S is a substring of a string belonging to C.*

Definition 3. *A C-compatible set S is a base (of C), if removing any element of S implies that the resulting set is not compatible.*

Thus, having in mind just compatibility, the bases are in this sense optimal.

Definition 4. *A segment set S is C-homogeneous if each element of C can be obtained uniquely as concatenation of the segments belonging to the set S.*

When we use the term homogenity (as distinct from C-homogenity), we mean, that the segment set is C-homogeneous, but we do not specify the related set C explicitly. Many real instances of segments databases in concatenative speech synthesis are homogeneous, for example, allophones, diphones, etc. A more complicated situation arises with segments sets based on syllables or syllables combined with morphemic segments (see e.g. [5, 7, 9]), where the homogenity depends on the concrete choice of the segment database.

Definition 5. *A C-compatible set S is strongly homogeneous if no element of S is a substring of a different element of S.*

Definition 6. *A compatible set S is strictly C-homogeneous if for any $u \in S$ do not exist $v \in S$ and $w \in S$ such that $v \neq u$, $w \neq u$ and u is a substring of vw.*

Definition 7. *A compatible set S is C-heterogeneous if it is not C-homogeneous.*

Let us recall some elementary properties of the basic classes of segments [8].

Proposition 1.

- *Any base is a consistent set.*
- *If S is strongly C-homogeneous, then it is C-homogeneous.*
- *If S is strictly C-homogeneous, then it is strongly C-homogeneous.*

3 Determining Homogenity and Bases of Homogeneous Sets

In this section we present an algorithm that can effectively decide whether a segment set is homogeneous. Then we show that for homogeneous segments we can easily determine their base.

Proposition 2. *Let S be a C-homogenous segment set and let $C_1 \subseteq C$. Then S is C_1 homogeneous.*

From this property it follows that if S is S^*-homogeneous then it is C-homogeneous for any C such that S is C-compatible. This raises a natural question. How to find out that a segment set is S^*-homogeneous? The following procedure represents a polynomial algorithm that decides whether or not a given segment set S is S^*-homogeneous.

Input: A segment set $S = \{s_1, \ldots, s_n\}$
Output: *YES* if S is S^*-homogenous, *NO* otherwise

```
1:  E := S;
2:  repeat
3:      Temp := ∅;
4:      for each v ∈ Suf(S) \ E do
5:          if there are s ∈ S and v' ∈ E such that
6:              v = sv' or s = vv'
7:          then Temp := Temp ∪ {v};
8:      E := E ∪ Temp;
9:  until Temp = ∅
10: if there are s, s' ∈ S such that s = s'v where v ∈ E
11:     then return NO;
12:     else return YES;
```

Fig. 1. An algorithm which decides whether or not S is S^*-homogenous.

Theorem 1. *The problem whether or not a given segment set S is S^*-homogeneous is decidable in polynomial time.*

Proof. Let $S = \{s_1, \ldots, s_n\}$ be a segment set. An *index sequence* (for S) is a finite sequence $\alpha = \alpha_1, \cdots, \alpha_k$ of natural numbers, where $k \geq 1$ is the *length* of α (denoted $length(\alpha)$), and $1 \leq \alpha_i \leq n$ for every α_i. Each such α determines a unique word $w_\alpha = s_{\alpha_1} s_{\alpha_2} \cdots s_{\alpha_k}$ over the alphabet A.

The set of all suffixes of segments in S is denoted $Suf(S)$. More precisely, $Suf(S) = \{v \mid \exists s \in S, v' \in A^* : s = v'v\}$. Note that $S \subseteq Suf(S)$, and the size of $Suf(S)$ is $\mathcal{O}(m^2)$ where m is the total length of all strings in S. We say that $v \in Suf(S)$ is *erasable* if there are index sequences α, β such that $vw_\alpha = w_\beta$.

We claim that S is S^*-heterogeneous if and only if there are two different $s, s' \in S$ such that $s = s'v$ and v is erasable. The '$\Longleftarrow$' direction is obvious. For the other direction, it suffices to realize that if S is S^*-heterogeneous, there must be (by definition) two

different index sequences γ, δ such that $w_\gamma = w_\delta$. First, realize that we can safely remove the longest common prefix of γ, δ from these sequences; the resulting sequences γ', δ' still have the property $w_{\gamma'} = w_{\delta'}$. Since $\gamma'_1 \neq \delta'_1$, $s_{\gamma'_1} \neq s_{\delta'_1}$ and hence one of the two segments must be a proper prefix of the other (otherwise it could not be that $w_{\gamma'} = w_{\delta'}$). Let us assume that, e.g., $s_{\gamma'_1}$ is a prefix of $s_{\delta'_1}$. That is, $s_{\gamma'_1} v = s_{\delta'_1}$ for some non-empty suffix v of $s_{\delta'_1}$. Hence, if we define α and β to be the sequences obtained by deleting the first element of γ' and δ', respectively, we have that $v w_\alpha = w_\beta$. This means that $v \in \mathit{Suf}(S)$ is erasable and we are done.

So, to determine whether or not S is S^*-homogenous, it suffices to compute the set $E \subseteq \mathit{Suf}(S)$ of all erasable suffixes and then check whether there are $s, s' \in S$ and $v \in E$ such that $s = s'v$. We prove that the set E is computed by the lines 1–9 of the algorithm presented in Fig. 1. First, realize that

'the variable E contains only strings which are erasable suffixes'

is an invariant of the loop presented in lines 2–9 (this follows immediately by inspecting the **if** statement in lines 5–7). Since E is initialized to S (line 1) and all segments of S are erasable suffixes, this invariant is surely valid before the first iteration of the loop. Hence, it is also satisfied after the last iteration (note that the total number of iterations of the loop is $\mathcal{O}(m^2)$). It remains to show that every erasable suffix eventually appears in the set assigned to the variable E. To do that, for every erasable suffix v we define its *norm*, denoted $norm(v)$, as follows:

$$norm(v) = \min\{\mathit{length}(\alpha) + \mathit{length}(\beta) \mid v w_\alpha = w_\beta\}$$

We claim that after the i^{th} iteration of the loop (lines 2–9), the variable E contains all erasable suffixes whose norm is at most $i + 1$. This is easy to verify by induction on i — since $norm(v) = 1$ for every $v \in S$, the induction base is established; and the induction step is proven easily by inspecting the code in lines 5–6 (the induction hypothesis is used to determine the content of E at line 5).

The previous observations imply the correctness of the algorithm in Fig. 1. The fact that it requires only polynomial time is also obvious (we only need to realize that the loop in lines 2–9 terminates after $\mathcal{O}(m^2)$ iterations, where m is the size of S). □

If we have a C-compatible segment set S, our interest is in reducing it as much as possible, simultaneously preserving compatibility. In other words, we would like to find out a C-base that is included in S, which has minimal number of segments. In general, we do not know a polynomial algorithm to solve this problem. However, we present a solution to this problem for homogeneous segments.

Proposition 3. *Let S be a C-homogeneous segment set. Then there exists precisely one C-base, which is a subset of S. The following procedure is a polynomial algorithm that determines the C-base:*

1. i := 0; $X := \emptyset$;
2. i := i + 1;
3. Determine the segmets $s_1, \ldots, s_k$ such that $s_1 \ldots s_k = u(i)$ ($u(i)$ being the i-th element of C) and add them to the set X.

4. **if** i does not exceed the number of the elements of C, **go to** 2
5. Remove all the elements that do not belong to X from S. This reduced set S' is a C-base (and it is a subset of S).

Proof. The proof easily follows from the fact that the segments $u_1, \ldots, u_k$ with the property $u_1 \ldots u_k = s(i)$ are determined uniquely. Hence, they cannot be removed from S. On the other hand, if a segment is not used for concatenating an element from C, it can be removed. The polynomiality of the algorithm follows directly from its description; in fact, the algorithm is linear with respect to the number of elements in C. □

4 Finding the Segment Representation of a Word for Homogeneous Segment Set

Finding a segment representation is a very natural problem. For concatenative speech synthesis, this task is usually a real-time problem which has to be performed quickly. Hence, effective algorithms solving this problem are of practical importance. A polynomial algorithm solving a more general problem can be found in [2, 3]. If we restrict ourselves to homogeneous segments, we can get a quicker and simplier algorithm, which is presented in what folows.

Problem 1. Let S be a homogeneous segment set. Let $t \in S^*$. The task is to find the sequence $(s_1, s_2, \ldots, s_n)$ of elements of S such that $t = s_1 s_2 \ldots s_n$.

The following algorithm finds, given a homogeneous segment set S and $t \in S^*$ a solution to Problem 1.

1. $P_0 := \{(u) | u \in S;$ there exists $v \in A^*$ such that $t = uv\}$.
2. i := 0
3. **while** no element of P_i is a solution to the problem **do**
   (a) $P_{i+1} := \{(s_1, s_2 \ldots, s_k, u) |\ (s_1, s_2, \ldots, s_k) \in P_i,\ u \in S$; exists $v \in A^*$ such that $t = s_1 s_2 \ldots s_k uv\}$.
   (b) $i := i + 1$
4. **return** the element of P_i, which is a solution to the problem (there is only one).

The algorithm constructs step by step all possible sequences of elements of S. As $t \in L(S)$ there must exist a sequence $(s_1, s_2, \ldots, s_n)$ such that $t = s_1 s_2 \ldots s_n$. Thus, after a finite number of steps the algorithm finds this sequence.

Theorem 2. *Let us denote by $K = \max_{x \in S} |x|$ and by n the length of t. The time complexity of the above algorithm is $O(Kn)$.*

Proof. If $s_1, s_2, \ldots, s_k$ is an element of P_i for some i, it cannot be an element of any P_j for any $j \neq i$. Indeed, as S is homogeneous, the string $s_1 s_2 \ldots s_k$ (a prefix of t) can be formed by no concatenation of elements of S other than $s_1, s_2, \ldots, s_k$. But, there are only n different prefixes of t. Thus, the total number of elements of all P_i during the whole run of the algorithm is not larger than n.

An effective implementation can ensure that the construction of an element of P_{i+1} given an element of P_i requires the time proportional to the length of the newly added string (if we assume that the time complexity of determining whether a string u is an element of S is $O(|u|)$). Thus, the time complexity of the algorithm is $O(Kn)$. □

As was mentioned, Problem 1 can be generalized to heterogeneous sets. In this case, the task is to find a sequence of strings ensuring a minimal number of concatenations. The time complexity of the general problem is $O(K^2n^2)$ (see [3]).

5 Conclusions and Future Work

The results presented in the paper show that homogeneous segment sets possesses some properties that can be advantageous in using them and processing them. In practice we often meet segment sets, that are "nearly homogeneous", which means that the homogenity is violated for a small number of cases. Investigation of this situation may be valuable from a practical point of view and is the main objective for future research.

Acknowledgement

The authors are grateful to James Thomas for proofreading a draft of the paper. The research has been partially supported by Czech Ministry of Education under the research project CEZ:J07/98:143300003.

References

1. E.C. Albano, P.A. Aquino: Linguistic Criteria for Building and Recording Units for Concatenative Speech Synthesis in Brazilian Portuguese, in Proceedings of Eurospeech, Rhodes, Greece, pp. 725-728, 1997.
2. R. Batůšek. An objective measure for assessment of the concatenative tts segment inventories. In *Proceedings of Eurospeech 2001 — Scandinavia*, Aalborg, Denmark, Sept. 2001.
3. R. Batůšek. *Speech Segments and Their Applications in Natural Language Processing*. PhD thesis, Masaryk University, Brno, Czech Republic, 2003. to be defended.
4. S. Deligne, F. Bimbot: Inference of Variable-Length Linguistic and Acoustic Units by Multigrams, Speech Communication 23 (1997), 223-241.
5. G. Doddington: Syllable Based Speech Processing; WS97 Project Report, Research Notes No. 30, J. Hopkins University, 1997.
6. A.J. Hunt, A.W. Black: Unit Selection in A Concatenative Speech Synthesis System Using a Large Database, in Proceedings of ICSLP, Philadelphia, pp. 373-376, 1996.
7. L. Josifovski, D. Mihajlov, D. Gorgevik: Speech Synthesizer Based on Time Domain Syllable Concatenation; Proceedings SPECOM'97, Cluj-Napoca, 1997, pp. 165-170.
8. I. Kopeček. Algebraic models of speech segment databases. In *Proceedings of TSD 2001*, pages 208–213, Železná Ruda, Czech Republic, 2001.
9. I. Kopeček: Automatic Segmentation into Syllable Segments; Proceedings of First Int. Conference on Language Resources and Evaluation, 1998, pp. 1275-1279.
10. Jon R. W. Yi and James R. Glass. Natural-sounding speech synthesis using variable-length units. In *The 5th International Conference on Spoken Language Processing*, Sydney, Australia, 1998.