Audio Identication using N-grams

Martin Møller Larsen
Anders Skovsgaard
Lars Tabro Sørensen

Outline

- Motivation
- Demonstration
- Related Work
- Fingerprint Technique
- The Proposed Algorithm
- Test Results
- Next Semester
- Questions

Motivation

- Compact string fingerprint suited for mobile devices.
- Generic backend for fingerprint searching.
- Audio Identification using small sample data.
- Fast and precise database lookup.

Demonstration

Related Work

- Sony/Ericsson TrackID
- Levenshtein distance (fuzzy searching)
- Spell checkers
- Suffix tree
  - Longest common substring
- DNA searching techniques.
  - Also represented as strings.
  - Is too strict.
  - BLAST
  - SSAHA – hashing, n-grams

Fingerprint

- For each song we got we create fingerprint
- The steps in the FutureProofFingerPrintFunction:
  - Decodes the audio to PCM and converts it to mono
  - Then it is resampled to sample rate of 8 KHZ
  - A low-pass filter is used on the input signal
  - The input signal is analyzed in pieces of 500 ms
  - A fast Fourier Transformation is to get the frequency domain

Fingerprint

- When the frequency domain is analyzed we generate a symbol each 62,5 ms from the codebook
- The codebook contains the vectors which corresponds to a specific frequency domain

The Proposed Algorithm

Based on the SSAHA but with character position threshold.

Example:

Building index:

s = EHFFNNFFNN9FFNNFFvvvvQ/rrrttAZvNNtttAAA99Q

The Proposed Algorithm

s = **EH**FFNNFFNN9FFNNFFvvvvQ/rrrttAZvNNtttAAA99Q

Index:

EH	Song METADATA Pos: o

The Proposed Algorithm

s = E**HF**FNNFFNN9FFNNFFvvvvQ/rrrttAZvNNtttAAA99Q

Index:

EH	Song METADATA Pos: 0
HF	Song METADATA Pos: 1

The Proposed Algorithm

s = EH**FF**NNFFNN9FFNNFFvvvvQ/rrrttAZvNNtttAAA99Q

Index:

EH	Song METADATA Pos: 0
HF	Song METADATA Pos: 1
FF	Song METADATA Pos: 2

The Proposed Algorithm

s = EHF**FN**NFFNN9FFNNFFvvvvQ/rrrttAZvNNtttAAA99Q

Index:

EH	Song METADATA Pos: 0
HF	Song METADATA Pos: 1
FF	Song METADATA Pos: 2
FN	Song METADATA Pos: 3
…	…

The Proposed Algorithm

Example:

Searching the database:

s = EHFFNNFFNN9FFNNFFvvvvQ/rrrttAZvNNtttAAA99Q
q = N9FFNFF

The Proposed Algorithm

s = EHFFNNFFNN9FFNNFFvvvvQ/rrrttAZvNNtttAAA99Q
q = **N9**FFNFF

Gram	Pos in q	Pos in s	Group
N9	0	9	9

The Proposed Algorithm

s = EHFFNNFFNN9FFNNFFvvvvQ/rrrttAZvNNtttAAA99Q
q = N**9F**FNFF

Gram	Pos in q	Pos in s	Group
N9	0	9	9
9F	1	10	9

The Proposed Algorithm

s = EHFFNNFFNN9FFNNFFvvvvQ/rrrttAZvNNtttAAA99Q

q = N9**FF**NFF

Gram	Pos in q	Pos in s	Group
N9	0	9	9
9F	1	10	9
FF	2	2 6 11 15	0 4 9 13

The Proposed Algorithm

s = EHFFNNFFNN9FFNNFFvvvvQ/rrrttAZvNNtttAAA99Q

q = N9F**FN**FF

Gram	Pos in q	Pos in s	Group
N9	0	9	9
9F	1	10	9
FF	2	2 6 11 15	0 4 9 13
FN	3	3 7 12	0 4 9

The Proposed Algorithm

s = EHFFNNFFNN9FFNNFFvvvvQ/rrrttAZvNNtttAAA99Q

q = N9FF**NF**F

Gram	Pos in q	Pos in s	Group
N9	0	9	9
9F	1	10	9
FF	2	2 6 11 15	0 4 9 13
FN	3	3 7 12	0 4 9

Gram	Pos in q	Pos in s	Group
NF	4	5 14	1 10

The Proposed Algorithm

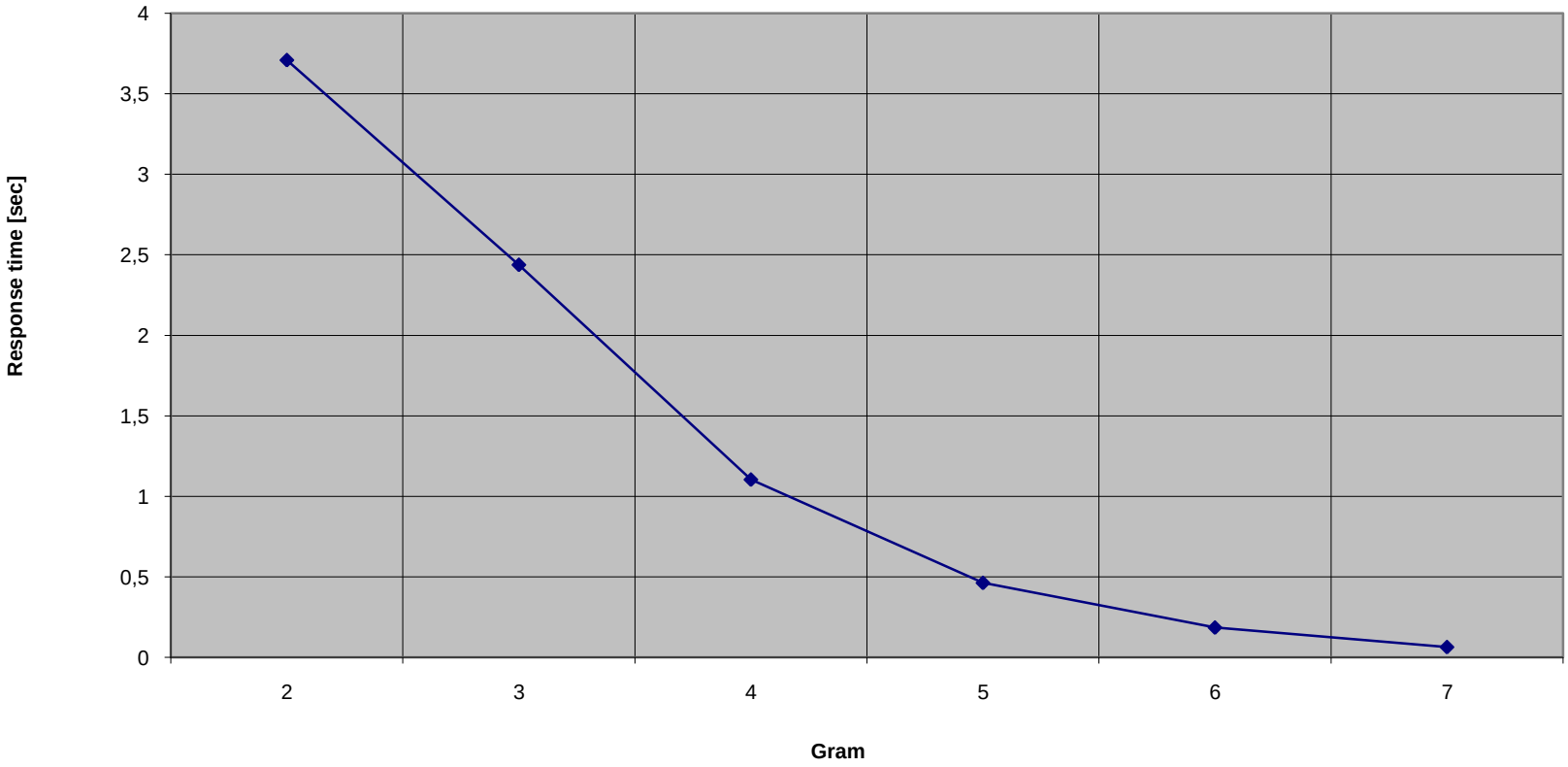
s = EHFFNNFFNN9FFNNFFvvvvQ/rrrttAZvNNtttAAA99Q

q = N9FFN**FF**

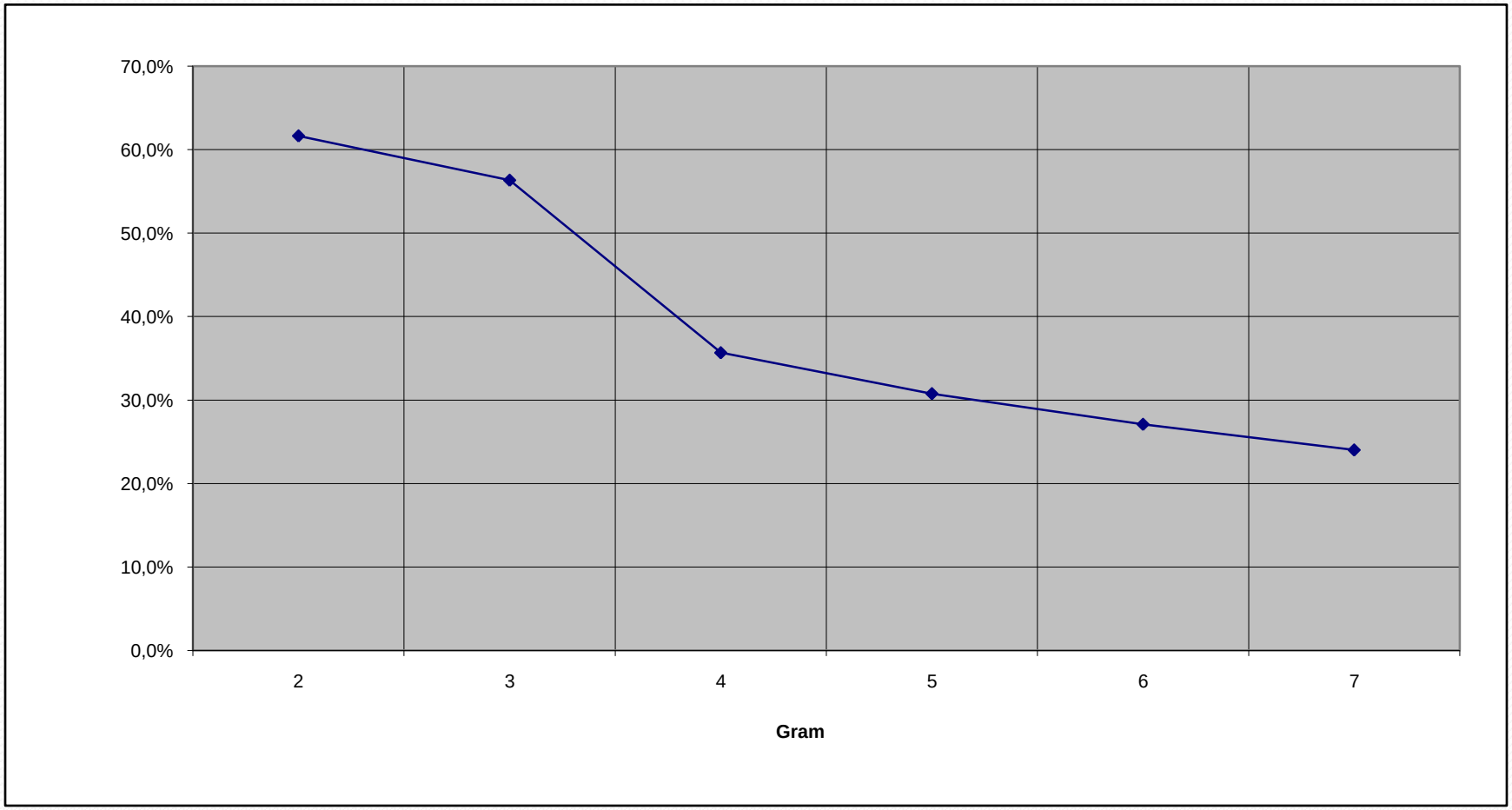
Gram	Pos in q	Pos in s	Group
N9	0	9	9
9F	1	10	9
FF	2	2 6 11 15	0 4 9 13
FN	3	3 7 12	0 4 9

Gram	Pos in q	Pos in s	Group
NF	4	5 14	1 10
FF	5	2 6 11 15	-3 1 6 10

Test Results



Test Results



Next Semester

- Improve the algorithm so that it can search for similar string using techniques like levenshtein
- Optimize the system for even larger database with more songs

Questions

1. Is it possibly to define a relationship between our output symbols from our codebook in order to use the levenshtein technique?

2. How can the fingerprint system be improved so it is more robust regarding distorted sound recording?

3. Is there more relevant performance studies?