Computational Linguistics at CIS - from basic research to applications in Digital Humanities

Stefan Gerdjikov, Klaus U. Schulz
CIS - LMU Munich

Including joint work with Stoyan Mihov and Petar Mitankin (Sofia)
Survey of talk

Excursion 1

Fast approximate search in large lexica
Interactive postcorrection of OCRd historical texts
Advanced text index structures

Excursion 2

Advanced text index structures
Modernization/normalization of historical language
Novel ways of querying and exploring corpora and text collections

Loose ends...

Ontologies & finding topics in texts (TopicZoom)
Wittgenstein search (Max Hadersbeck)
Latin (Uwe Springmann, Helmut Schmidt)
Excursion 1

Fast approximate search in large lexica

Interactive postcorrection of OCRed historical texts

Advanced text index structures
Approximate search in lexica

Intuition
Given a large lexicon L and possibly erroneous input tokens, for each input token V efficiently find the „most similar“ entries W of L.

Levenshtein distance d
The Levenshtein distance between two words V, W is the minimal number of edit operations (deletion, insertion of a symbol or substitution of one symbol by another) needed to rewrite V into W.

Formalized problem
Given a lexicon L, an input word („pattern“) V and a bound b (=0,1,2,..), compute all entries W of L such such d(V,W) ≤ b.
Lexicon representation as minimal deterministic finite-state automaton

Example lexicon: axis, behave, child, church, cold, hate, hold
Lexicon representation as minimal deterministic finite-state automaton

Example lexicon: axis, behave, child, church, cold, hate, hold
Lexicon representation as minimal deterministic finite-state automaton

Example lexicon: axis, behave, child, church, cold, hate, hold

☐ Fast access
☐ Size: each prefix of a lexicon word only represented once. Similar sharing of suffixes!
Lexicon representation as minimal deterministic finite-state automaton

Example lexicon: axis, behave, child, church, cold, hate, hold

- Fast access
- Size: each prefix of a lexicon word only represented once. Similar sharing of suffixes!
- Using the lexicon: for many "processing tasks" strings occurring in several words only processed once.
Approximate search in lexicon using dynamic programming (Oflazer)

Search for „chiid“, maximal Levenshtein distance 1
Approximate search in lexicon using dynamic programming (Oflazer)

Search for „chiid“, maximal Levenshtein distance 1

„Complete“ traversal of lexicon automaton

Parallel control
Oflazer approximate search

Search for "chiid", maximal Levenshtein distance 1
Oflazer approximate search

Search for „chiid“, maximal Levenshtein distance 1

Blind path, backtracking!
Oflazer approximate search

Search for “chiid“, maximal Levenshtein distance 1
Oflazer approximate search

Search for „chiid“, maximal Levenshtein distance 1
Oflazer approximate search

Search for "chiid", maximal Levenshtein distance 1
Oflazer approximate search

Search for „chiid“, maximal Levenshtein distance 1
Oflazer approximate search

Search for “chiid“, maximal Levenshtein distance 1

match: child
Oflazer approximate search

For given pattern,

- “blind“ paths in automaton recognized as early as possible,
- only “small“ part of the automaton visited

But

- Computing matrix columns at each step expensive!
Approximate search using non-deterministic Levenshtein automata

Search for "chiid", maximal Levenshtein distance 1

active states ↓↓
Approximate search using non-deterministic Levenshtein automata

Search for „chiid“, maximal Levenshtein distance 1
Approximate search using non-deterministic Levenshtein automata

Search for „chiid“, maximal Levenshtein distance 1

Transition with x fails - backtracking
Approximate search using non-deterministic Levenshtein automata

Search for "chiid", maximal Levenshtein distance 1
Approximate search using non-deterministic Levenshtein automata

Search for "chiid", maximal Levenshtein distance 1
Approximate search using non-deterministic Levenshtein automata

Search for „chiid“, maximal Levenshtein distance 1
Approximate search using non-deterministic Levenshtein automata

Search for „chiid“, maximal Levenshtein distance 1
Approximate search using non-deterministic Levenshtein automata

Search for „chiid“, maximal Levenshtein distance 1

match: child
Approximate search using non-deterministic Levenshtein automata

Same strengths

But similar weaknesses:

- Specific automaton for each pattern,
- Propagation of active states computationally expensive.
Approximate search using non-deterministic Levenshtein automata

Observation 1

The set of active states after step n always restricted to a “triangular area“.
Approximate search using non-deterministic Levenshtein automata

Observation 1

The set of active states after step $n$ always restricted to a „triangular area“. 
Approximate search using non-deterministic Levenshtein automata

**Observation 1**

The set of active states after step n always restricted to a „triangular area“. 
Approximate search using non-deterministic Levenshtein automata

Observation 1

The set of active states after step n always restricted to a „triangular area“.
Approximate search using non-deterministic Levenshtein automata

Observation 1

The set of active states after step n always restricted to a „triangular area“. 
Approximate search using non-deterministic Levenshtein automata

Observation 1

The set of active states after step n always restricted to a „triangular area“.
Approximate search using non-deterministic Levenshtein automata

Observation 1

The set of active states after step $n$ always restricted to a “triangular area“.
Approximate search using non-deterministic Levenshtein automata

Observation 2

Given any subset of active states in the triangular area and an input symbol $\sigma$, the next set of active states only depends on the question: which labels of horizontal transitions in the triangular area are identical to $\sigma$?
Determinization / Universal Automata

Use “abstracted“ triangular areas with distributions of active states (without any reference to an input string) as states of a universal deterministic automaton. Bitvectors used as input encode transition info.

„Universal deterministic Levenshtein automaton“ (here: bound $b=1$)
Example: input “chiid”, b=1

Figure 6
The universal deterministic Levenshtein automaton $A^*(1)$. See Example 4 for notation.
Example: input “chiid”, b=1

Figure 6
The universal deterministic Levenshtein automaton $A^\gamma(1)$. See Example 4 for notation.
Example: input “chiid”, $b=1$

\[ \chi(a,\$chi)=0000 \]
Example: input “chiid”, b=1
Example: input “chiid”, b=1

\[ \chi(x, \text{chii})=0000 \]
Example: input “chiid”, b=1

\[ \chi(x, \text{chii}) = 0000 \]

No transition with 0000
Example: input “chiid”, b=1

\[ \chi(c, \$\chi) = 0100 \]
Example: input “chiid”, b=1
Example: input “chiid”, b=1

\[ \chi(h, chii) = 0100 \]
Example: input “chiid”, b=1

Figure 6
The universal deterministic Levenshtein automaton $A^\gamma(1)$. See Example 4 for notation.
Example: input “chiid”, b=1

\[\chi(i, \text{hiid})=0100\]
Example: input “chiid”, b=1

Figure 6
The universal deterministic Levenshtein automaton $A^i(1)$. See Example 4 for notation.
Example: input “chiid”, b=1

\[ \chi(l, iid) = 000 \]
Example: input “chiid”, b=1
Example: input “chiid”, b=1

\[ \chi(d, id) = 01 \]
Example: input “chiid”, b=1

Output „child“

Figure 6
The universal deterministic Levenshtein automaton $A^Y(1)$. See Example 4 for notation.
Advantages

- For each bound $n=1,2,3,4$ universal Levenshtein automaton offline precomputed once.

- Computing bitvectors and tracing states in the universal Levenshtein automaton cheap!!

Very fast control mechanism for traversal of lexicon automaton. Useful solution for approximate search in a large lexicon (Lucene).
but not fully satisfactory...

Checked transitions

Figure 6
The universal deterministic Levenshtein automaton $A^Y(1)$. See Example 4 for notation.
... better solution ...

Assume we would know that the there cannot be an error in the first part „chi“ of pattern „chiid“ ........

Checked transitions
... better solution ...

Assume we would know that the there cannot be an error in the first part „chi“ of pattern „chiid“……..

Transitions to be checked
... better solution ...

Error somewhere in pattern

---------------------------------

(a) Error in second half of pattern

---------------------------------
... better solution ...

Error somewhere in pattern
????????????????????
---------------------------------

(a) Error in second half of pattern
√ ???????????
---------------------------------

(b) Error in first half of pattern
??????????? √ ?
---------------------------------
... better solution ...

Error somewhere in pattern

(a) Error in second half of pattern

(b) Error in first half of pattern

Use an automaton for inverted lexicon words for backward search!
... better solution ...

Error somewhere in pattern

(a) Error in second half of pattern

(b) Error in first half of pattern

Use an automaton for inverted lexicon words for backward search!
Forward-backward method

Can be generalized to larger bounds n=2, 3,…

\[
\begin{array}{c|c|c}
0 & \leq 1 & \leq 1 \\
\hline
n=1 & \rightarrow & \leftarrow \\
\hline
\leq 1 & \leq 1 \ (2) & \leq 1 \ (2) \\
\hline
n=2 & \rightarrow & \leftarrow \\
\hline
\leq 1 & \leq 2 & \leq 2 \\
\hline
n=3 & \rightarrow & \leftarrow \\
\end{array}
\]

Enormous speed-up factor! Fully satisfactory solution for

- small bounds (1,2,3,4)
- normal lexica (entries not too long)
Postcorrection of OCRRed historical Texts
Postcorrection of OCRred historical Texts

Given a historical text obtained via Optical Character Recognition (OCR)

- How can we found tokens that „probably“ represent errors?
- How can we find correction suggestions – also in original historical spelling?
- How can we find corresponding modern words?
- How can we find typical OCR errors and errors series?
- How can we find characteristics of historical orthography?
Two-channel model for OCRed historical texts

OCR ed historical text

.....tnurm......
................
.jeshs chr1stus
................
lcyd............
Two-channel model for OCRed historical texts

OCR channel

Historical text (ground truth)  OCRed historical text

.....thurm......
............
.jesus christus
............
leyd............

OCR errors

.....tnurm......
............
.jeshs chrlestus
............
lcyd............

h->n
u->h
i->l
e->c
Two-channel model for OCRed historical texts

OCR channel

Modern form

.....turm........
............... 
jesus christus
............... 
leid...........

Historical text (ground truth)

t-> th

ei-> ey

OCRRed historical text

.....tnurm......
.................
jesus christus
................. 
leyd...........

OCR errors

h->n

u->h

i-> l

e->c
Two-channel model for OCRed historical texts

Historical language channel

Modern form

......turm........
.................
jesus christus
...............
leid...........

OCR channel

Historical text (ground truth)

......thurm.....
................
.jesus christus
............... 
leyd...........

OCRed historical text

......tnurm.....
................
jeshs chrlstus
............... 
lcyd...........

„historical patterns“

OCR errors

t-> th

h->n

u->h

i-> l

e->c
Computing all „interpretations“ of OCR token

„Interpretation“: starting from OCR token $W_{ocr}$ „guess“ channel history

Modern word form

$W_{mod}$

\[ ? \]

word form in ground truth

$W_{gt}$

\[ ? \]

word form in OCRed text

$W_{ocr}$

Patterns applied „pattern trace“

OCR errors applied „OCR trace“

Variant of forward-backword procedure:
Given lexicon of modern words, set of patterns, set of possible OCR errors.
Efficiently generate all interpretations for an OCRed word
(number of pattern applications and OCR errors bounded)
Profiling historical OCRed corpora

for each OCR token $W_{ocr}$

Initial „rough“ model for
• words
• patterns
• OCR errors
and their probabilities
Profiling historical OCRed corpora

for each OCR token $W_{ocr}$

- generate ranked list of interpretations with probabilities

- Modern word
- Hist trace
- Ground truth
- OCR trace

Initial „rough“ model for
- words
- patterns
- OCR errors and their probabilities
Profiling historical OCRed corpora

for each OCR token $W_{ocr}$

Initial „rough“ model for
• words
• patterns
• OCR errors
and their probabilities

Summing up relative frequencies over all OCR tokens obtain
Profiling historical OCRed corpora

for each OCR token $W_{ocr}$

First improved model for
- words
- patterns
- OCR errors
and their probabilities

Summing up relative frequencies over all OCR tokens obtain
Profiling historical OCRed corpora

for each OCR token $W_{ocr}$

generate ranked list of interpretations with probabilities

First improved model for
- words
- patterns
- OCR errors and their probabilities

Iterate - after n rounds obtain
Profiling historical OCRed corpora

for each OCR token $W_{ocr}$

- Improved list of interpretations with probabilities
- Improved model for:
  - words
  - patterns
  - OCR errors and their probabilities

Modern word

Hist trace

Ground truth

OCR trace

Local guess

Global guess

Final Result
PoCoTo Postcorrection Tool

Profiles used for interactive postcorrection of OCRed historical texts

- finding tokens that „probably“ represent errors
- finding correction suggestions, also in original historical spelling
- finding corresponding modern words (not realized)

- finding typical OCR errors and errors series
- finding characteristics of historic orthography (not realized)
Deutschland und Belgien
ΤΟΥΣ, ΚΑΙ ΝΕΡΑ \ν
ΚΑΙ ΤΗΝ ΓΕΝΗ, ΚΑΙ ΤΗΝ ΓΕΝΗ.
ΚΑΙ ΘΕΟΤΕΡΑΣ, ΚΑΙ ΘΕΟΤΕΡΑΣ.
ΠΑΝ ΧΛΩΡΟΝ ΆΓΡΟΥ, ΠΑΝ ΧΛΩΡΟΝ ΆΓΡΟΥ.
ΕΓΕΝΟΝΤΟ ΠΑΣΑΙ ΑΙ
ΥΨΧΗΣΙ ΟΥ ΦΑΓΕΣΘΕ, ΥΨΧΗΣΙ ΟΥ ΦΑΓΕΣΘΕ.
ΓΑΡ ΤΟ ΨΗΤΕΡΟΝ, ΓΑΡ ΤΟ ΨΗΤΕΡΟΝ.
ΤΗΣ ΓΗΣ, ΤΗΣ ΓΗΣ.
ΚΑΤΕΘΙ ΚΥΡΙΟΣ ΙΔΕΙΝ
ΕΞΗΛΘΕΝ ΕΚ ΧΑΡΡΑΥ, ΕΞΗΛΘΕΝ ΕΚ ΧΑΡΡΑΥ.
ΤΟ ΏΡΝΟΜΑ ΚΥΡΙΟΥ, ΤΟ ΏΡΝΟΜΑ ΚΥΡΙΟΥ.
ΧΩΤΩ ΣΥΜΠΟΡΕΟΜΕΝΩ, ΧΩΤΩ ΣΥΜΠΟΡΕΟΜΕΝΩ.

ΙΧ 3

ΓΕΝΕΣΙΣ

χειρας υμων δεσωκα, Και παν έρπετον ο εστιν ζων υμων εσται εις βαζω

σιν ος λαχανα χρωτου έδωκα υμων τα παντα. Πλην κρας εν αιματι

ψυχης ου φαγεσθε, Και γαρ το υμετερον αιμα των ψυχων υμων έκης

τηςω. Εκ χειρων παυσων των θηριων έκησησαι αυτο, και εκ χειρων

Approximate search
...the story continued......
Approximate search
...the story continued......

Forward-backward

\[
\begin{align*}
\text{n=1} & \quad 0 \leq 1 \quad \leftrightarrow \quad 1 \leq 0 \\
\text{n=2} & \quad 0 \leq 2 \quad \leftrightarrow \quad 2 \leq 0 \\
\text{n=3} & \quad 0 \leq 3 \quad \leftrightarrow \quad 3 \leq 0 \\
\text{n=4} & \quad 0 \leq 4 \quad \leftrightarrow \quad 4 \leq 0
\end{align*}
\]
Approximate search
...the story continued......

If a pattern containing $\leq n$ errors is split into $n+1$ parts, one part has 0 errors!!

?Generalization of forward-backward where search can start with any infix of the pattern?

Need a data structure for the lexicon that
1. gives immediate access to any infix
2. infixes can be extended both to the left and to the right
Approximate search
...the story continued......

Using „symmetric compacted directed acyclic word graphs SCDAWGs“:

Improved algorithm for approximate search in lexica and collections of strings - much better for longer strings and larger distance bounds.

„Wallbreaker“ system – winner of international competition 2013 for string similarity search.

......Insight

SCDAWGs and related text index structures very interesting for many natural language processing tasks!!