Computing Locally Coherent Discourses

Alexander Koller
Saarland University

joint work with
Ernst Althaus and Nikiforos Karamanis

PARC
19 October 2004
Overview

- Local coherence and the Discourse Ordering Problem
- Some examples for local coherence measures
- Equivalence of Discourse Ordering and TSP
- Computing discourse orderings
- Evaluation
Local coherence

- Human-written text is not a random ordering of discourse units.
- Discourse units are ordered to maximise coherence.
- Local coherence: Coherence is measured in terms of unit-to-unit transition costs.
Local coherence

Discourse ordering problem:
Find best ordering according to local coherence

- many, many
- other orderings
Local coherence: Our definition (2-place)

1. Initial cost $c_i$: discourse starts with this unit

2. Transition costs $c_T$: costs of unit-to-unit transitions

3. Initial cost $c_i$: discourse starts with this unit

transition costs $c_T$: costs of unit-to-unit transitions
Local coherence: Our definition (3-place)

1. Initial cost $c_i$: discourse starts with these two units.

2. Transition costs $c_T$: depend on two preceding units.

3. Diagram showing the flow of discourse with the initial and transition costs.
The Discourse Ordering Problem

- d-place Discourse Ordering Problem

- **Input:**
  - discourse units $u_1, \ldots, u_n$
  - d-place transition costs $c_T$
  - (d-1)-place initial costs $c_I$

- **Output:**
  - a permutation $\pi$ of $\{1, \ldots, n\}$ such that
    \[
    c_f(u_{\pi(1)}, \ldots, u_{\pi(d-1)})
    \]
    \[
    + \sum_{i=1}^{n-d+1} c_T(u_{\pi(i+d-1)}|u_{\pi(i)}, \ldots, u_{\pi(i+d-2)})
    \]
    is minimal.
Some cost functions from the literature

- Based on Centering Theory:
  - coherence of a transition is defined in terms of entity coherence
  - Karamanis & Manurung 02, Karamanis et al. 04 compare such cost functions

- Based on statistical models:
  - Lapata 03 gives cost function based on various features of adjacent sentences
  - \( d = 2, c_T(u_2 \mid u_1) = - \log P(u_2 \mid u_1) \)
The entities referred to in a discourse unit are called the forward-looking centers.

Cf(u) is the list of entities in unit u, ranked by salience.

Cp(u) -- the preferred center -- is the highest-ranked member of Cf(u).

Cb(u_i) -- the backward-looking center -- is the highest-ranked member of Cf(u_i) that also appears in Cf(u_{i-1}).
## CT-based transitions and cost functions

<table>
<thead>
<tr>
<th></th>
<th>COHERENCE:</th>
<th>COHERENCE*:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\text{Cb}(u_i) = \text{Cb}(u_{i-1})$</td>
<td>$\text{Cb}(u_i) \neq \text{Cb}(u_{i-1})$</td>
</tr>
<tr>
<td><strong>SALIENCE:</strong></td>
<td>CONTINUE</td>
<td>SMOOTH-SHIFT</td>
</tr>
<tr>
<td>$\text{Cb}(u_i) = \text{Cp}(u_i)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SALIENCE</strong>*:</td>
<td>RETAIN</td>
<td>ROUGH-SHIFT</td>
</tr>
<tr>
<td>$\text{Cb}(u_i) \neq \text{Cp}(u_i)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Method      | $d$ | Initial cost $c_I(u_1, \ldots, u_{d-1})$ | Transition cost $c_T(u_d|u_1, \ldots, u_{d-1})$ |
|-------------|-----|------------------------------------------|-----------------------------------------------|
| M.NOCB      | 2   | 0                                        | $\text{nochb}_2$                             |
| M.KP        | 3   | $\text{nochb}_2 + \text{nocheap}_2 + \text{nosal}_2$ | $\text{nochb}_2 + \text{nocheap}_2 + \text{nosal}_2 + \text{noch}_3$ |
| M.BFP       | 3   | $1 - \text{nosal}_2, \text{nosal}_2, 0, 0$ | $(\text{cont}_3, \text{ret}_3, \text{ss}_3, \text{rs}_3)$ |
Computational issues

- Number of permutations is too big for generate-and-test. Need 77 years for discourse of length 20.

- How hard is the problem really?

- If it is hard, can we solve it anyway?
  - Mellish et al. 98: Genetic Programming
  - Lapata 03: Approximative graph algorithm
  - No guarantees for quality of found solution
Discourse ordering as a graph problem

- each edge has a cost
- find cheapest path that visits all nodes

new start node
The Discourse Ordering Problem (with graphs)

- **Input:**
  - Graph $G = (V, E)$
  - Start node $s \in V$
  - $d$-place cost function $c : V^d \to \mathbb{R}$

- **Output:**
  - A simple directed path $P = (s = v_0, v_1, \ldots, v_n)$ that visits all vertices, such that
    \[
    \sum_{i=0}^{n-d+1} c(v_i, v_{i+1}, \ldots, v_{i+d-1})
    \]
    is minimal.
Travelling Salesman Problem

- Find cheapest round trip in a graph that visits every node exactly once.
- Classical NP-complete problem.
- Cannot be approximated: There is no general polynomial algorithm that guarantees good solutions (unless P = NP).
- Generalised asymmetric TSP (GATSP): Partition nodes into disjoint sets, tour must visit each set exactly once.
d-PDOP is equivalent to TSP

- Can reduce TSP to 2-PDOP:
  - hence NP-hard,
    no approximation algorithms

- Can reduce d-PDOP to GATSP:
  - hence, can apply algorithms for TSP
  - need GATSP to encode d-place cost function for \( d > 2 \).
Reduction of TSP to 2-PDOP

TSP

2-PDOP
Reduction of d-PDOP to GATSP

GATSP: Visit every column exactly once
Solving GATSP by Linear Programming

- A standard method for solving TSP and other combinatorial problems: Integer Linear Programming (ILP).
- Write problem as set of linear equations and inequalities.
- Find integer numbers that satisfy all (in)equations.
A Linear Program for GATSP

\[
\begin{align*}
\min & \quad \sum_{e \in E} c_e x_e \\
\text{s.t.} & \quad \sum_{e \in \delta^+(v)} x_e = \sum_{e \in \delta^-(v)} x_e \quad \forall v \in V \quad (1) \\
& \quad \sum_{e \in \delta^-(V_i)} x_e = 1 \quad 1 \leq i \leq n \quad (2) \\
& \quad \sum_{e \in \delta^+(\bigcup_{i \in I} V_i)} x_e \geq 1 \quad I \subset \{1, \ldots, n\} \quad (3) \\
& \quad x_e \in \{0, 1\}
\end{align*}
\]
Solving the Linear Program

- **Branch-and-cut technique**
- Solve linear (in)equations for arbitrary (not necessarily integer) values.
- Pick variable $x$ with fractional value and solve subproblems with $x = 1$ and $x = 0$.
- Continue until all variables have integer values.
- Use heuristics to pick good variable and keep search space small.
Further Optimisations

- Exponential number of inequalities of Type 3:
  - start with a small subset of inequalities
  - check preliminary solutions for violations of Type 3 inequalities
  - add violated inequalities by need

- Add redundant inequalities that can be violated by non-integer solutions.
Evaluation

- Four cost functions:
  - M.LAPATA: $d = 2$, statistics-based
    (evaluated on discourses from BLLIP corpus)
  - Three centering-based cost functions
    (evaluated on discourses from GNOME corpus)

- Random graphs of different sizes

- Two ILP solvers:
  - CPLEX 9.0 (commercial)
  - SCIP 0.6.3 / SOPLEX 1.2.2a (free)
Evaluation: \( d = 2 \)

<table>
<thead>
<tr>
<th>Instance</th>
<th>Size</th>
<th>ILP-FS</th>
<th>ILP-CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>lapata-10</td>
<td>13</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>coffers1 M.NOCB</td>
<td>10</td>
<td>0.04</td>
<td>0.02</td>
</tr>
<tr>
<td>cabinet1 M.NOCB</td>
<td>15</td>
<td>0.07</td>
<td>0.01</td>
</tr>
<tr>
<td>random (avg)</td>
<td>20</td>
<td>0.09</td>
<td>0.07</td>
</tr>
<tr>
<td>random (avg)</td>
<td>40</td>
<td>0.28</td>
<td>0.17</td>
</tr>
<tr>
<td>random (avg)</td>
<td>60</td>
<td>1.39</td>
<td>0.40</td>
</tr>
<tr>
<td>random (avg)</td>
<td>100</td>
<td>6.17</td>
<td>1.97</td>
</tr>
</tbody>
</table>

(seconds CPU time, Pentium 4 at 3 GHz)
## Evaluation: $d = 3$

<table>
<thead>
<tr>
<th>Instance</th>
<th>Size</th>
<th>ILP-FS</th>
<th>ILP-CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>coffers1 M.KP</td>
<td>10</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>coffers1 M.BFP</td>
<td>10</td>
<td>0.08</td>
<td>0.06</td>
</tr>
<tr>
<td>cabinet1 M.KP</td>
<td>15</td>
<td>0.40</td>
<td>1.12</td>
</tr>
<tr>
<td>cabinet1 M.BFP</td>
<td>15</td>
<td>0.39</td>
<td>0.28</td>
</tr>
<tr>
<td>random (avg)</td>
<td>10</td>
<td>1.00</td>
<td>0.42</td>
</tr>
<tr>
<td>random (avg)</td>
<td>15</td>
<td>35.1</td>
<td>5.79</td>
</tr>
<tr>
<td>random (avg)</td>
<td>20</td>
<td>-</td>
<td>115.8</td>
</tr>
</tbody>
</table>

(seconds CPU time, Pentium 4 at 3 GHz)
Both cabinets probably entered England in the early nineteenth century / after the French Revolution caused the dispersal of so many French collections. / The pair to [this monumental cabinet] still exists in Scotland. / The fleurs-de-lis on the top two drawers indicate that [the cabinet] was made for the French King Louis XIV. / [It] may have served as a royal gift, / as [it] does not appear in inventories of [his] possessions. / Another medallion inside shows [him] a few years later. / The bronze medallion above [the central door] was cast from a medal struck in 1661 which shows [the king] at the age of twenty-one. / A panel of marquetry showing the cockerel of [France] standing triumphant over both the eagle of the Holy Roman Empire and the lion of Spain and the Spanish Netherlands decorates [the central door]. / In [the Dutch Wars] of 1672 - 1678, [France] fought simultaneously against the Dutch, Spanish, and Imperial armies, defeating them all. / [The cabinet] celebrates the Treaty of Nijmegen, which concluded [the war]. / The Sun King’s portrait appears twice on [this work]. / Two large figures from Greek mythology, Hercules and Hippolyta, Queen of the Amazons, representatives of strength and bravery in war appear to support [the cabinet]. / The decoration on [the cabinet] refers to [Louis XIV’s] military victories. / On the drawer above the door, gilt-bronze military trophies flank a medallion portrait of [the king].

(cabinet1, M.NOCB; cost = 2)
Evaluation: Interpretation

- Runtimes for $d = 2$ are fast.
- For $d = 3$, still quite fast for real-life examples.
- Coherence measures seem to be easier than random graphs.
Conclusion

- Discourse Ordering problem is equivalent to Travelling Salesman.

- Can use heuristics to compute optimal solution very efficiently (up to 50 discourse units per second).

- Applications:
  - real systems
    (generation and summarisation)
  - experimentation
Future Work

- Optimise GATSP algorithm.
- What makes real-life instances easier than random graphs?
- Global coherence measures (i.e., arrange units in discourse tree structure)?