Support Vector Machines and Kernel Methods for Co-Reference Resolution

Alessandro Moschitti and Xiaofeng Yang

2007 Summer Workshop on Human Language Technology
Center for Language and Speech Processing
John Hopkins University
Baltimore, August 22, 2007
Outline

- Motivations
- Support Vector Machines
- Kernel Methods
  - Polynomial Kernel
  - Sequence Kernels
  - Tree kernels
- Kernels for Co-reference problem
  - An effective syntactic structure
  - Mention context via word sequences
- Experiments
- Conclusions
Motivations

- Intra/Cross document coreference resolution require the definition of complex features, i.e.
  - syntactic/semantic structures

- For pronoun resolution
  - Preference factors: Subject, Object, First-Mention, Definite NP
  - Constraint factors: C-commanding,…

- For non-pronoun
  - Predicative Structure, Appositive Structure
Motivations (2)

- How to represent such structures in the learning algorithm?
- How to combine different features?
- How to select the relevant ones?
- Kernel methods allows us to
  - represent structures in terms of substructures (high dimensional feature spaces)
  - define implicit and abstract feature spaces
- Support Vector Machines “select” the relevant features
  - Automatic Feature engineering side-effect
Support Vector Machines

The margin is equal to \( \frac{2|k|}{\|w\|} \)

We need to solve

\[
\max \frac{2|k|}{\|w\|} \\
\vec{w} \cdot \vec{x} + b \geq +k, \text{ if } \vec{x} \text{ is positive} \\
\vec{w} \cdot \vec{x} + b \leq -k, \text{ if } \vec{x} \text{ is negative}
\]
SVM Classification Function and the Kernel Trick

- From the primal form

\[ f(\vec{x}) = \text{sgn}(\vec{x} \cdot \vec{w} + b) \]
SVM Classification Function and the Kernel Trick

- From the primal form

\[ f(\vec{x}) = \text{sgn}(\vec{x} \cdot \vec{w} + b) \]

- To the dual form

\[ f(\vec{x}) = \text{sgn} \left( \sum_{i=1}^{\ell} y_i \alpha_i \vec{x}_i \cdot \vec{x} + b \right) = \]

where \( \ell \) is the number of training examples
SVM Classification Function and the Kernel Trick

- From the primal form

$$f (\vec{x}) = \text{sgn}(\vec{x} \cdot \vec{w} + b)$$

- To the dual form

$$f (\vec{x}) = \text{sgn}\left( \sum_{i=1..\ell} y_i \alpha_i \vec{x}_i \cdot \vec{x} + b \right) = \text{sgn}\left( \sum_{i=1..\ell} y_i \alpha_i \phi(o_i) \cdot \phi(o) + b \right) = \text{sgn}\left( \sum_{i=1..\ell} y_i \alpha_i k(o_i, o) + b \right)$$

where $\ell$ is the number of training examples
Flat features (Linear Kernel)

- Documents in Information Retrieval are represented as word vectors

$$\vec{x} = (0, \ldots, 1, \ldots, 0, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1)$$

  buy acquisition stocks sell market

- The dot product $$\vec{x} \cdot \vec{z}$$ counts the number of features in common

- This provides a sort of similarity
**Feature Conjunction (polynomial Kernel)**

- The initial vectors are mapped in a higher space

\[
\Phi : \langle x_1, x_2, x_3 \rangle \rightarrow \langle x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3 \rangle
\]

\[
\Phi : \langle z_1, z_2, z_3 \rangle \rightarrow \langle z_1, z_2, z_3, z_1z_2, z_1z_3, z_2z_3 \rangle
\]

- \(\langle \text{Stock, Market, Downtown} \rangle \rightarrow \langle \text{Stock, Market, Downtown, Stock+Market, Downtown+Market, Stock+Downtown} \rangle\)

- We can efficiently compute the scalar product as

\[
K_{\text{Poly}}(\langle x_1, x_2, x_3 \rangle, \langle z_1, z_2, z_3 \rangle) = \Phi(\langle x_1, x_2, x_3 \rangle) \cdot \Phi(\langle z_1, z_2, z_3 \rangle) = \left(\langle x_1, x_2, x_3 \rangle \cdot \langle z_1, z_2, z_3 \rangle + 1 \right)^2
\]
String Kernel

- Given two strings, the number of matches between their substrings is evaluated

- E.g. Bank and Rank
  - B, a, n, k, Ba, Ban, Bank, Bk, an, ank, nk,..
  - R, a, n, k, Ra, Ran, Rank, Rk, an, ank, nk,..

- String kernel over sentences and texts

- Huge space but there are efficient algorithms
Word Sequence Kernel

- String kernels where the symbols are words
- e.g. “so Bill Gates says that” ⇒
  - Bill Gates says that
  - Gates says that
  - Bill says that
  - so Gates says that
  - so says that
  - ...

A Tree Kernel
[Collins and Duffy, 2002]
The overall SST fragment set
Explicit kernel space

\[ \vec{x} = (0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0, \ldots, 1, \ldots, 0) \]

- Given another vector \( \vec{z} \),
- \( \vec{x} \cdot \vec{z} \) counts the number of common substructures
Implicit Representation

\[
\bar{x} \cdot \bar{z} = \phi(T_x) \cdot \phi(T_z) = K(T_x, T_z) = \\
= \sum_{n_x \in T_x} \sum_{n_z \in T_z} \Delta(n_x, n_z)
\]

[Collins and Duffy, ACL 2002] evaluate \( \Delta \) in \( O(n^2) \):

\[
\Delta(n_x, n_z) = 0, \text{ if the productions are different else} \\
\Delta(n_x, n_z) = 1, \text{ if pre-terminals else} \\
\Delta(n_x, n_z) = \prod_{j=1}^{nc(n_x)} (1 + \Delta(ch(n_x, j), ch(n_z, j)))
\]
Kernels for Co-reference problem: Syntactic Information

- Syntactic knowledge is important
  - For pronoun resolution
    - Subject, Object, First-Mention, Definite NP, C-commanding, …?
  - For non-pronoun
    - Predicative Structure, Appositive Structure …

- Source of syntactic knowledge: Parse Tree:
  - How to utilize such knowledge…
Previous Works on Syntactic knowledge

Define a set of syntactic features extracted from parse trees
- whether a candidate is a subject NP
- whether a candidate is an object NP
- whether a candidate is c-commanding the anaphor
- ....

Limitations
- Manually design a set of syntactic features
- By linguistic intuition
- Completeness, Effectiveness?
Incorporate structured syntactic knowledge – main idea

- Use parse tree directly as a feature
- Employ a tree kernel to compare the similarity of the tree features in two instances
- Learn a SVM classifier
**Syntactic Tree feature**

- Subtree that covers both anaphor and antecedent candidate
  ⇒ syntactic relations between anaphor & candidate (subject, object, c-commanding, predicate structure)
- Include the nodes in path between anaphor and candidate, as well as their first_level children

— "the man in the room saw him"
— inst("the man", "him")
A word sequence representing the mention expression and its context
- Create a sequence for a mention

-“Even so, Bill Gates says that he just doesn’t understand our infatuation with thin client versions of Word ”
- (so), (Bill)(Gates)(says)(that)
Composite Kernel

- different kernels for different features
  - Poly Kernel: for baseline flat features
  - Tree Kernel: for syntax trees
  - Sequence Kernel: for word sequences
- A composite kernel for all kinds of features
- Composite Kernel = TreeK*PolyK+PolyK+SequenceK
Results for pronoun resolution

<table>
<thead>
<tr>
<th></th>
<th>MUC-6</th>
<th></th>
<th>ACE-02-BNews</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>P</td>
<td>F</td>
<td>R</td>
</tr>
<tr>
<td>All attribute value features</td>
<td>64.3</td>
<td>63.1</td>
<td>63.7</td>
<td>58.9</td>
</tr>
<tr>
<td>+Syntactic Tree + Word Sequence</td>
<td>65.2</td>
<td>80.1</td>
<td><strong>71.9</strong></td>
<td>65.6</td>
</tr>
</tbody>
</table>
## Results for over-all coreference Resolution using SVMs

<table>
<thead>
<tr>
<th></th>
<th>MUC-6</th>
<th></th>
<th></th>
<th>ACE02-BNews</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>P</td>
<td>F</td>
<td>R</td>
<td>P</td>
<td>F</td>
</tr>
<tr>
<td>BaseFeature SVMs</td>
<td>61.5</td>
<td>67.2</td>
<td>64.2</td>
<td>54.8</td>
<td>66.1</td>
<td>59.9</td>
</tr>
<tr>
<td>BaseFeature + Syntax Tree</td>
<td>63.4</td>
<td>67.5</td>
<td>65.4</td>
<td>56.6</td>
<td>66.0</td>
<td>60.9</td>
</tr>
<tr>
<td>BaseFeature+Syntax Tree + Word Sequences</td>
<td>64.4</td>
<td>67.8</td>
<td>66.0</td>
<td>57.1</td>
<td>65.4</td>
<td>61.0</td>
</tr>
<tr>
<td>All Sources of Knowledge</td>
<td>60.1</td>
<td>76.2</td>
<td>67.2</td>
<td>60.0</td>
<td>65.4</td>
<td>63.0</td>
</tr>
</tbody>
</table>
Conclusions

- SVMs and Kernel methods are powerful tools to design intra/cross doc coreference systems

- SVMs allows for
  - better exploit attribute/vector features
  - the use of syntactic structures
  - the use of word sequence context

- The results show noticeable improvement over the baseline
Thank you