Challenges and Solutions in Automatically Annotating CMC Data

Torsten Zesch
Senior Researcher
UKP Lab, Technische Universität Darmstadt

Associated Researcher
German Institute for International Educational Research, Frankfurt

International workshop:
Building Corpora of Computer-Mediated Communication: Issues, Challenges, and Perspectives
Feb 14–15, 2013
TU Dortmund
Challenging Properties of CMC Data

Very heterogeneous (Twitter, Wikipedia, Chat, Discussion forum, …)
- No tool fits all
- Even access might be challenging

One’s junk is the other’s research interest :) 
- Many common assumptions do not hold

Non-standard language
- “lmao s/o to the cool ass asian officer 4 #1 not runnin my license and #2 not takin dru boo to jail . Thank u God . #amen”
- Standard tools might not work & need to deal with spelling errors, abbreviations, and variants

Often (relatively) small samples
- Hard to build automatic models
Working with CMC Data

Mix of manual and automatic tasks
Wrong tool for the job?
Availability of Tools & Models
Proposed Solution

Comprehensive NLP Framework
Apache UIMA
Unstructured Information Management Architecture

- Component-based architecture
- Designed to scale
- Flexible type system
- High-density in-memory data representation

- 2004 – IBM alphaWorks project
  - still used e.g. in IBM LanguageWare
- 2006 – Apache Incubator project
- 2009 – OASIS Standard
- 2010 – Full Apache project
- 2010 – Used in IBM’s Watson
  Jeopardy Challenge

http://uima.apache.org
Pipeline Architecture

Reader

Linguistic preprocessing

Morphological analysis

Syntactic analysis

Semantic analysis

Writer

Stemmer

Lemmatizer

Compound splitter

PoS tagger

Parser

Tokenizer

Sentence splitter

Stopword tagger

Named entity tagger

Datastore / Results
Information Flow

Source Document Collection

Module_1

Documents → Tokens

Module_2

Documents → Tokens → Sentences

Module_3

Documents → Tokens → Sentences → POS

Annotated Document Collection
So far, so good … but the framework is empty
DKPro Framework

- DKPro Core
  - Linguistic preprocessing
  - Tokenization, tagging, parsing, co-reference, n-gram models
- Support for various file formats
  - XML, PDF, WSDL, Wikipedia, TEI, IMS-CWB, ANNIS, …
- Already available open source
- Coming soon
  - Information Retrieval, Keyphrase Extraction, Summarization, Lexical Cohesion
Darmstadt Knowledge Processing Software Repository

DKPro Core
- Tokenizer
- Part of Speech Tagger
- Lemmatizer
- Parser
- Named Entity Recognition
- I/O

DKPro Semantics
- Keyphrase Extraction
- Text Segmenting
- Link Discovery
- Text Summary
- Relatedness
- Spelling

DKPro Information Retrieval
- Indexing API
- Query API
- Apache Lucene
- Terrier 2
- Terrier 3
- Latent Semantic Indexing
Darmstadt Knowledge Processing Software Repository

DKPro Core
- Tokenizer
- Part of Speech Tagger
- Lemmatizer
- Parser
- Named Entity Recognition
- I/O

DKPro Semantics
- Keyphrase Extraction
- Text Segmenting
- Link Discovery
- Text Summary
- Relatedness
- Spelling

DKPro Information Retrieval
- Indexing API
- Query API
- Apache Lucene
- Terrier 2
- Terrier 3
- Latent Semantic Indexing
Multilinguality

- Automatically select the correct language-specific components
Other Dimensions

Flexibility
- Change I/O or replace tagger without breaking the pipeline

Scalability
- Process large amounts of data on Hadoop cluster

Efficiency
- Integrated access to large resources, e.g. web-scale language models

Robustness & Accuracy
- Not influenced by the choice of the framework
- But: UIMA encourages to improve existing tools instead of developing new ones
Proposed Solution

Comprehensive NLP Framework

Pro
- Large collection of interoperable components

Contra
- Programming knowledge needed (only very little)
- Not (mainly) targeted towards CMC
Challenge

Cool Tool – Wrong Environment
Domain Adaptation / Retraining
Examples

Different smiley styles :) :-) (^_^) ^o #smiley

<table>
<thead>
<tr>
<th>Different smiley styles</th>
<th>: ) : - ) ( ^ _ ^ ) ^ o # smiley</th>
</tr>
</thead>
<tbody>
<tr>
<td>JJ JJ NN NNS : ) : : ) ( SYM SYM SYM ) SYM NN # NN</td>
<td></td>
</tr>
</tbody>
</table>

KEEP UP ON YOUR READING WITH AUDIO BOOKS

Audio books are highly popular with library patrons in the town of Springfield, Greene County, MO. "People are mobile and busier, and audio books fit into that lifestyle" says Gary Sanchez, who oversees the library's $2 million budget...
Proposed Solution

Annotate & Retrain
Learning Curves

Collins-CFG parser

LTAG Parser

Figure 3: The learning curve for the Collins-CFG parser in terms of F-scores for increasing amounts of manually annotated training data. Performance for sentences ≤ 40 words is plotted.

Figure 4: The learning curve for the LTAG parser in terms of F-scores for increasing amounts of training data. Performance when evaluated on sentences of length ≤ 40 words is plotted.

Case Study
Retraining Dependency Parsers for German

All the work done by Irina Alles
- Joint supervision with Chris Biemann

4 Parsers
- MaltParser
- MateTools Parser
- Stanford Parser
- MDParse (DKFI inhouse)
Case Study
Retraining Dependency Parsers for German

All the work done by Irina Alles
  ▪ Joint supervision with Chris Biemann

4 Parsers
  ▪ MaltParser
  ▪ MateTools Parser
  ▪ Stanford Parser
  ▪ MDParser (DKFI inhouse)
Case Study
Retraining Dependency Parsers for German

All the work done by Irina Alles
  • Joint supervision with Chris Biemann

4 Parsers
  • MaltParser
  • MateTools Parser
  • Stanford Parser
  • MDParser (DKFI inhouse)

2 annotated corpora
  • TIGER Treebank (50,000 sentences of German newspaper text)
  • TueBa-D/Z treebank (65,524 sentences of German newspaper text)

MaltEval using CoNLL evaluation format (LA – labelled accuracy)
Training Time
Testing Time
Parsing Accuracy

![Bar chart showing parsing accuracy for different models: Malt TiGER, Malt Tueba, Mate TiGER, Mate Tueba, MDP TiGER. Each model has three accuracy levels: 10.00%, 50.00%, and 100.00%. The values are: Malt TiGER 89.00%, Malt Tueba 93.90%, Mate TiGER 88.20%, Mate Tueba 85.50%, MDP TiGER 85.50%.](chart.png)
Parsing Accuracy – Ensemble Parser

![Bar chart showing parsing accuracy for Coverage and Accuracy with different percentages for 10.00%, 50.00%, and 100.00%.]
Proposed Solution

Annotate & Retrain

Pro
- Often remarkably few training instances are necessary

Contra
- Re-trainability of tools needs to be improved
- Some tools behave weird when trained on data that is too different

Like CMC data 😊
I am in a special situation ...
Proposed Solution

Build special tools
Proposed Solution

Build special tools

Enable researchers to customize tools

Retraining
- Tool stays exactly the same – new behavior due to new training data

Customization
- Customize tool when retraining is not enough
Use Case – Twitter Tagger

- Mixture of **customized tool** and **retraining**
- Customized tokenizer
- Retrained tagger
Use Case – Twitter Tagger

Different smiley styles :) :-) (^_^) ^o #smiley

<table>
<thead>
<tr>
<th>Different</th>
<th>smiley styles</th>
<th>:)</th>
<th>:-)</th>
<th>(^</th>
<th>^</th>
<th>^</th>
<th>^o</th>
<th>#smiley</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A N E E E E EMO EMO HASH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADJ</td>
<td>ADJ NN EMO EMO EMO E E #</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Different</th>
<th>smiley styles</th>
<th>:)</th>
<th>:-)</th>
<th>(^</th>
<th>_</th>
<th>^</th>
<th>^</th>
<th>^</th>
<th>^</th>
<th>^o</th>
<th>#</th>
<th>smiley</th>
</tr>
</thead>
<tbody>
<tr>
<td>JJ</td>
<td>NN NNS : ) : : ) ( SYM SYM SYM ) SYM NN # NN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADJ</td>
<td>NN NN O O O O O O O O O O O O NN O NN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Different smiley styles :) :-) (^_^) ^o #smiley

Different Tokenization

<table>
<thead>
<tr>
<th>Different</th>
<th>smiley</th>
<th>styles</th>
<th>:)</th>
<th>:-)</th>
<th>(^</th>
<th>^)</th>
<th>^o</th>
<th>#smiley</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>N</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>EMO</td>
<td>EMO</td>
<td>HASH</td>
</tr>
<tr>
<td>ADJ</td>
<td>ADJ</td>
<td>NN</td>
<td>EMO</td>
<td>EMO</td>
<td>EMO</td>
<td>E</td>
<td>E</td>
<td>#</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Different</th>
<th>smiley</th>
<th>styles</th>
<th>: )</th>
<th>: - )</th>
<th>( ^ _ ^ )</th>
<th>^ o</th>
<th>#</th>
<th>smiley</th>
</tr>
</thead>
<tbody>
<tr>
<td>JJ</td>
<td>NN</td>
<td>NNS</td>
<td>: )</td>
<td>: : )</td>
<td>( SYM SYM SYM )</td>
<td>SYM</td>
<td>NN</td>
<td>#</td>
</tr>
<tr>
<td>ADJ</td>
<td>NN</td>
<td>NN</td>
<td>O O O O O O O O O O O O O O O O NN O</td>
<td>NN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Use Case – Twitter Tagger

Different smiley styles :) :-) (^_^) ^o #smiley

<table>
<thead>
<tr>
<th>Different smiley styles</th>
<th>:)</th>
<th>:-)</th>
<th>(^</th>
<th>^</th>
<th>^</th>
<th>^o</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>N</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>EMO EMO</td>
</tr>
<tr>
<td>ADJ</td>
<td>ADJ</td>
<td>NN</td>
<td>EMO</td>
<td>EMO</td>
<td>EMO</td>
<td>E</td>
</tr>
</tbody>
</table>

Specialized Tagset.

<table>
<thead>
<tr>
<th>Different smiley styles</th>
<th>:)</th>
<th>:-)</th>
<th>(^</th>
<th>_</th>
<th>^</th>
<th>^</th>
<th>^o</th>
</tr>
</thead>
<tbody>
<tr>
<td>JJ</td>
<td>NN</td>
<td>NNS</td>
<td>(</td>
<td>SYM</td>
<td>SYM</td>
<td>SYM</td>
<td>(</td>
</tr>
<tr>
<td>ADJ</td>
<td>NN</td>
<td>NN</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

#smiley HASH #
Use Case – Twitter Tagger
Retrained Model

Spending the day withhh mommma !

<table>
<thead>
<tr>
<th>Spending</th>
<th>the</th>
<th>day</th>
<th>withhh</th>
<th>mommma</th>
<th>!</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>D</td>
<td>N</td>
<td>P</td>
<td>N</td>
<td>,</td>
</tr>
</tbody>
</table>

Twitter Tokenizer + Twitter Tagger

<table>
<thead>
<tr>
<th>Spending</th>
<th>the</th>
<th>day</th>
<th>withhh</th>
<th>mommma</th>
<th>!</th>
</tr>
</thead>
<tbody>
<tr>
<td>VVG</td>
<td>DT</td>
<td>NN</td>
<td>NN</td>
<td>NNS</td>
<td>SENT</td>
</tr>
</tbody>
</table>

Standard Tokenizer + TreeTagger
Use Case – Twitter Tagger
Retrained Model

Spending the day withhh mommma !

Different Tagsets

<table>
<thead>
<tr>
<th>Spending</th>
<th>the</th>
<th>day</th>
<th>withhh</th>
<th>mommma</th>
<th>!</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>D</td>
<td>N</td>
<td>P</td>
<td>N</td>
<td>,</td>
</tr>
</tbody>
</table>

Twitter Tokenizer + Twitter Tagger

<table>
<thead>
<tr>
<th>Spending</th>
<th>the</th>
<th>day</th>
<th>withhh</th>
<th>mommma</th>
<th>!</th>
</tr>
</thead>
<tbody>
<tr>
<td>VVG</td>
<td>DT</td>
<td>NN</td>
<td>NN</td>
<td>NNS</td>
<td>SENT</td>
</tr>
</tbody>
</table>

Standard Tokenizer + TreeTagger
Use Case – Twitter Tagger
Retrained Model

Spending the day withhh mommma!

<table>
<thead>
<tr>
<th>Spending</th>
<th>the</th>
<th>day</th>
<th>withhh</th>
<th>mommma</th>
<th>!</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>D</td>
<td>N</td>
<td>P</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

Twitter Tokenizer + Twitter Tagger

<table>
<thead>
<tr>
<th>Spending</th>
<th>the</th>
<th>day</th>
<th>withhh</th>
<th>mommma</th>
<th>!</th>
</tr>
</thead>
<tbody>
<tr>
<td>VVG</td>
<td>DT</td>
<td>NN</td>
<td>NN</td>
<td>NNS</td>
<td>SENT</td>
</tr>
</tbody>
</table>

Better model → Better Tagging

Standard Tokenizer + TreeTagger
Use Case – Twitter Tagger
Retrained Model

Spending the day with hh mommma!

Mapping to generalized POS models

<table>
<thead>
<tr>
<th>Spending</th>
<th>the</th>
<th>day</th>
<th>with hh</th>
<th>mommma</th>
<th>!</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>D</td>
<td>N</td>
<td>P</td>
<td>N</td>
<td>,</td>
</tr>
<tr>
<td>V</td>
<td>ART</td>
<td>NN</td>
<td>PP</td>
<td>NN</td>
<td>PUNC</td>
</tr>
</tbody>
</table>

Twitter Tokenizer + Twitter Tagger

<table>
<thead>
<tr>
<th>Spending</th>
<th>the</th>
<th>day</th>
<th>with hh</th>
<th>mommma</th>
<th>!</th>
</tr>
</thead>
<tbody>
<tr>
<td>VVG</td>
<td>DT</td>
<td>NN</td>
<td>NN</td>
<td>NNS</td>
<td>SENT</td>
</tr>
<tr>
<td>V</td>
<td>ART</td>
<td>NN</td>
<td>NN</td>
<td>NN</td>
<td>PUNC</td>
</tr>
</tbody>
</table>

Standard Tokenizer + TreeTagger
Proposed Solution

Build special tools

Enable researchers to customize tools

Pro
- Very flexible

Contra
- Still too complicated for non programmers

Work in progress ...
Summary

Availability of Tools & Models
- DKPro framework

Domain Adaptation / Retraining
- a little annotation can get you a long way

Customization
- active field of research in our Lab
- interesting research project opportunities