Mavuno: A Scalable and Effective Hadoop-Based Paraphrase Acquisition System

Donald Metzler and Eduard Hovy
Information Sciences Institute
University of Southern California
Overview

• Mavuno
  • Paraphrases 101
  • Large-scale paraphrase mining with Hadoop

• Evaluation
  • Paraphrasing verb phrases
  • Noisy text paraphrases (Twitter)

• Other Applications

• Conclusions
Paraphrases 101

• What is a paraphrase?
  • A way of saying the same thing, but in a different way.
  • An alternative way of stating the same thing.

• What are paraphrases useful for?
  • Information extraction, textual entailment, information retrieval, ...

• Paraphrase tasks
  • Detection (is sentence B a paraphrase of sentence A?)
  • Generation (given a sentence, build a paraphrased version of it)
  • Acquisition (identify paraphrases for an input phrase)

• Approaches
  • Distributional similarity (monolingual corpus)
  • Machine translation-based pivoting (parallel corpus)
• Mavuno is a scalable open-source Hadoop-based distributional similarity engine

• Design principles
  • Scalable
  • Modular
  • Flexible (researcher-friendly)

• How scalable?
  • Biggest experiment so far mined paraphrases from over 4.5TB of Web page data using a relatively small cluster (around 100 cores).
  • Could easily scale well beyond...
• Distributional hypothesis
  • Words (or phrases) that occur in the same contexts tend to have similar meanings

• Distributional similarity
  • Represents words (or phrases) in a vector space.
  • Each dimension of the vector space corresponds to some context.
  • Vector components are weighted according to the strength of the association between the word (or phrase) and the context.
  • Similarity between vectors is used as proxy for semantic relatedness.

• Model options
  • Context type
  • Weighting scheme
  • Similarity measure
• Choice of context is important, as it governs the nature of the vector space within which similarity is computed

• Tradeoffs
  • Simpler contexts are crude, require no NLP resources, and yield “dense” contextual representations.
  • More complex contexts capture meaning better, but require more NLP resources (e.g., POS tagger, chunker, parser), and yield more “sparse” contextual representations.

“George Bush campaigned in a snow-bound condominium complex Friday.”
• Mavuno makes use of Map/Reduce to compute distributional similarity at scale.

• Computing distributional similarity with Map/Reduce requires the following components:
  • Given a phrase, find all of the contexts it occurs in.
  • Given a context, find all of the phrases that occur in it.
  • Compute phrase/context association strengths.
  • Compute vector/vector similarities.

• Boils down to sparse matrix multiplication.
System Architecture

Document Corpus

Phrase -> Context Module

(input phrase₁, ... input phraseₙ)

(phrase, score₁, ... (phrase, scoreₙ)

Context -> Phrase Module

(context, score₁, ... (context, scoreₙ)
Algorithm 1 “phrase → context” mapper algorithm.

\[
ed \leftarrow \text{extractor} \\
d \leftarrow \text{input document} \\
P \leftarrow \text{input phrases} \\
W \leftarrow \text{input phrase weights} \\
T \leftarrow e.\text{extract}(d) \\
\text{for } (p, c) \in T \text{ do} \\
\quad \text{if } P.\text{contains}(p) \text{ then} \\
\qquad \text{emit}(c, p, W.\text{get}(p)) \\
\quad \text{end if} \\
\text{end for}
\]

- Given a set of (weighted) input phrases, identifies all of the contexts that the phrases occur in.
- Maps over the document corpus.
- Assumes that phrase \rightarrow weight hashtable fits in main memory.
Variable Length \textit{n}-gram Extractor

Algorithm 1 Pasca and Dienes extractor.

\begin{verbatim}
c_{\text{min}} \leftarrow \text{minimum context length}
c_{\text{max}} \leftarrow \text{maximum context length}
p_{\text{max}} \leftarrow \text{maximum phrase length}
d \leftarrow \text{input document}
T \leftarrow [ ]
\textbf{for} \ pos = 1 \ \textbf{to} \ \text{length}(d) \ \textbf{do}
\quad \textbf{for} \ plen = 1 \ \textbf{to} \ p_{\text{max}} \ \textbf{do}
\quad \quad \textbf{for} \ clen = c_{\text{min}} \ \textbf{to} \ c_{\text{max}} \ \textbf{do}
\quad \quad \quad p \leftarrow \text{getNgram}(d, pos, plen)
\quad \quad \quad c_l \leftarrow \text{getNgram}(d, pos - clen, clen)
\quad \quad \quad c_r \leftarrow \text{getNgram}(d, pos + plen, clen)
\quad \quad \quad c \leftarrow (c_l, c_r)
\quad \quad \text{T.add}((p, c))
\quad \end{verbatim}

\textbf{return} \ T

\textbf{Recommended settings:}
\begin{itemize}
\item \(c_{\text{min}} = 2, c_{\text{max}} = 3, p_{\text{max}} = 5\)
\end{itemize}
Algorithm 1 “phrase → context” reducer algorithm.

\[
c ← \text{input context} \\
T ← \text{input tuples } \{(p, w)\} \\
score ← 0 \\
\text{for } (p, w) \in T \text{ do} \\
\quad \text{score } ← \text{score } + \ s(c; p) \cdot w \\
\text{end for} \\
\text{emit}(c, \ score)
\]

- For each key (i.e., a context \(c\)), compute an aggregate score across the phrases that occur in \(c\)
- Reduces over the output of the mapper.
- A context is assigned a high score if it occurs in many highly weighted phrases that have large \(s(c; p)\) values.
Scoring Functions

- Scoring functions \( s(c; p) \) and \( s(p; c) \) measure the strength of association between a phrase and a context.
  - Many possible ways to compute these scores.
  - Typically defined symmetrically (i.e., \( s(c; p) = s(p; c) \)), but this is not necessary.

- "Overlap" scoring function

\[
s(p; c) = s(c; p) = \begin{cases} 
1 & \text{if } \#(c, p) > 0 \\
0 & \text{otherwise}
\end{cases}
\]

- Cosine similarity (pointwise mutual information weighting)

\[
s(p; c) = s(c; p) = \frac{pmi(p,c)}{\sqrt{\sum_{c' \in c(p)} pmi^2(p,c')}}
\]
• Everything is defined exactly as the phrase -> context case, except the role of phrases and contexts are swapped.

• Mapper emits (phrase, context, weight) tuples.

• Reducer aggregates and scores phrases.

• Natural symmetry between phrase -> context module and context -> phrase module allows chaining of operations.
Putting It All Together

• Acquiring paraphrases with Mavuno
  • **Input:** set of phrases to be paraphrased.
  • **Step 1:** Run input phrases through phrase -> context module.
  • **Step 2:** Run output of phrase -> context module through context -> phrase module.
  • **Step 3 (Optional):** Go to Step 1.
  • **Output:** Return the output of the context -> phrase module.

• Mathematical interpretation (sparse matrix multiplication)

\[
S_{p \rightarrow p} = S_{p \rightarrow c} S_{c \rightarrow p}
\]

\[
s(p'; p) = \sum_{c \in C(p)} s(p'; c)s(c; p)
\]
Evaluation

- Acquiring paraphrases for verb phrase chunks
  - Clean data
  - Wide variety of contexts can, and have, been used (e.g., \(n\)-grams, POS tags, parse tree paths, etc.)
  - Paraphrase methods range from simple and scalable (e.g., distributional similarity) to sophisticated and resource intensive (e.g., requiring

- Mining paraphrases (and lexical variants) from Twitter
  - Noisy data
  - Only simple \(n\)-gram contexts can be used
  - Very little (if any) existing research
Paraphrasing Verb Phrase Chunks

• Data
  • Randomly sampled 100 verb phrases from a news corpus.

• Methodology
  • Users were presented with a pair of sentences.
  • Sentence 1: A sentence that contains one of the original verb phrases.
  • Sentence 2: The same sentence as 1, with the original verb phrase substituted for a system-generated paraphrase.
  • Annotators judged if the two sentences were semantically equivalent.

• Annotation
  • Made use of Amazon’s Mechanical Turk service.
  • Employed a number of tactics to reduce spam and low quality work.
  • Gathered a total of 18,150 annotations (post-filtering).
  • Minimum of 1, median of 3, and max of 6 annotations per sentence pair.
  • Data is publically available.
<table>
<thead>
<tr>
<th>Sentence Pair</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A five-man presidential council for the independent state newly pro-</td>
<td>0</td>
</tr>
<tr>
<td>claimed in south Yemen was named overnight Saturday, it <strong>was officially</strong></td>
<td></td>
</tr>
<tr>
<td><strong>announced</strong> in Aden.</td>
<td></td>
</tr>
<tr>
<td>A five-man presidential council for the independent state newly pro-</td>
<td></td>
</tr>
<tr>
<td>claimed in south Yemen was named overnight Saturday, it <strong>was cancelled</strong></td>
<td></td>
</tr>
<tr>
<td>in Aden.</td>
<td></td>
</tr>
<tr>
<td>Dozens of Palestinian youths held rally in the Abu Dis Arab village in</td>
<td>1</td>
</tr>
<tr>
<td>East Jerusalem <strong>to protest</strong> against the killing of Sharif.</td>
<td></td>
</tr>
<tr>
<td>Dozens of Palestinian youths held rally in the Abu Dis Arab village in East</td>
<td></td>
</tr>
<tr>
<td>Jerusalem <strong>in protest of</strong> against the killing of Sharif.</td>
<td></td>
</tr>
<tr>
<td>It says that foreign companies have no greater right to compensation –</td>
<td>2</td>
</tr>
<tr>
<td><strong>establishing</strong> debts at a 1/1 ratio of the dollar to the peso – than</td>
<td></td>
</tr>
<tr>
<td>Argentine citizens do.</td>
<td></td>
</tr>
<tr>
<td>It says that foreign companies have no greater right to compensation –</td>
<td></td>
</tr>
<tr>
<td><strong>setting</strong> debts at a 1/1 ratio of the dollar to the peso – than</td>
<td></td>
</tr>
<tr>
<td>Argentine citizens do.</td>
<td></td>
</tr>
</tbody>
</table>
Summary of Results

- Brief summary of results (since you’re probably more interested in the large-scale aspects of the system...)
  - Simple approaches (e.g., using n-gram contexts, no NLP resources, etc.) obtain **comparable precision** and **better coverage** than sophisticated NLP-rich approaches.
  - Most paraphrase systems produce highly redundant results (e.g., “runs”, “ran”, “running”, “has run”, etc. for the input “went running”), which can make precision values look artificially high.
  - There is still considerable room for improvement, since even the best systems achieve rather poor accuracy levels.

- See the paper (and our ACL ’11 poster) for complete details
TREC12
- 800K news articles
- 2.2 GB of text

Gigaword
- 6.9M news articles
- 21GB of text

ClueWeb
- 50M Web pages
- 1.5TB of text

Bigger (data) isn’t always better!
**Mining Paraphrases from Twitter**

<table>
<thead>
<tr>
<th>abt → about</th>
<th>r → are</th>
<th>ru → en</th>
<th>b4 → before</th>
<th>btw → btwn</th>
</tr>
</thead>
<tbody>
<tr>
<td>chk → check</td>
<td>cld → could</td>
<td>deets → details</td>
<td>fab → great</td>
<td>fb → facebook</td>
</tr>
<tr>
<td>fav → fave</td>
<td>4 → 3</td>
<td>fyi → what a farce</td>
<td>fwd → forward</td>
<td>gr8 → great</td>
</tr>
<tr>
<td>ic → express</td>
<td>itz → syrenz vlog</td>
<td>jk → just kidding</td>
<td>jsyk → brusday</td>
<td>mil → corners on the fly</td>
</tr>
<tr>
<td>k → ck</td>
<td>omw → on my way</td>
<td>1 → 2</td>
<td>ppl → people</td>
<td>plz → please</td>
</tr>
<tr>
<td>shld → shud</td>
<td>tx → thx</td>
<td>da → the</td>
<td>2 → to</td>
<td>yr → year</td>
</tr>
</tbody>
</table>

- Gathered a list of terms/phrases commonly shortened or abbreviated in character-limited communication media, such as SMS and Twitter.

- Mined paraphrases from a large collection of Twitter data using Mavuno (variable length $n$-gram contexts, cosine scoring).

- Results in table show the top result for each input (bold indicates a correct pair).

- Simple techniques are effective, even for very noisy data.
## Mining Lexical Variants From Half a Billion Tweets

<table>
<thead>
<tr>
<th>(A)</th>
<th>(B)</th>
<th>(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>i ⇔ you</td>
<td>u ⇔ you</td>
<td>ur ⇔ your</td>
</tr>
<tr>
<td>my ⇔ the</td>
<td>seeking ⇔ seeks</td>
<td>wit ⇔ with</td>
</tr>
<tr>
<td>u ⇔ you</td>
<td>2 ⇔ to</td>
<td>to ⇔ too</td>
</tr>
<tr>
<td>is ⇔ was</td>
<td>lost ⇔ won</td>
<td>goin ⇔ going</td>
</tr>
<tr>
<td>a ⇔ the</td>
<td>q ⇔ que</td>
<td>kno ⇔ know</td>
</tr>
<tr>
<td>i ⇔ we</td>
<td>f*ck ⇔ hell</td>
<td>about ⇔ bout</td>
</tr>
<tr>
<td>my ⇔ your</td>
<td>feat ⇔ ft</td>
<td>wat ⇔ what</td>
</tr>
<tr>
<td>and ⇔ but</td>
<td>bday ⇔ birthday</td>
<td>jus ⇔ just</td>
</tr>
<tr>
<td>seeking ⇔ seeks</td>
<td>ff ⇔ followfriday</td>
<td>talkin ⇔ talking</td>
</tr>
<tr>
<td>me ⇔ you</td>
<td>yang ⇔ yg</td>
<td>gettin ⇔ getting</td>
</tr>
<tr>
<td>2 ⇔ to</td>
<td>wit ⇔ with</td>
<td>doin ⇔ doing</td>
</tr>
<tr>
<td>am ⇔ was</td>
<td>a ⇔ my</td>
<td>so ⇔ soo</td>
</tr>
<tr>
<td>are ⇔ were</td>
<td>are ⇔ r</td>
<td>you ⇔ your</td>
</tr>
<tr>
<td>lost ⇔ won</td>
<td>amazing ⇔ awesome</td>
<td>dnt ⇔ dont</td>
</tr>
<tr>
<td>he ⇔ she</td>
<td>til ⇔ till</td>
<td>bday ⇔ birthday</td>
</tr>
<tr>
<td>q ⇔ que</td>
<td>fav ⇔ favorite</td>
<td>nothin ⇔ nothing</td>
</tr>
<tr>
<td>it ⇔ that</td>
<td>mostly ⇔ partly</td>
<td>people ⇔ ppl</td>
</tr>
<tr>
<td>f*ck ⇔ hell</td>
<td>northbound ⇔ southbound</td>
<td>lil ⇔ little</td>
</tr>
<tr>
<td>can ⇔ could</td>
<td>hung ⇔ toned</td>
<td>sayin ⇔ saying</td>
</tr>
<tr>
<td>im ⇔ its</td>
<td>love ⇔ miss</td>
<td>so ⇔ sooo</td>
</tr>
</tbody>
</table>
Other Applications

- **Information extraction**
  - Mining relation patterns
  - Class instance mining

- **Web site/page similarity**
  - Site (or page) = “phrase”
  - Incoming links, anchor text, or clicks = “contexts”

- **Query expansion**
  - Terms = “phrase”
  - Document that one or more query terms occurs in = “context”
  - Weighting incorporates relevance of document to the query.
Conclusions

• Mavuno
  • Scalable Hadoop-based distributional similarity engine.
  • Highly configurable.
  • Useful for a variety of data mining, IR, and NLP applications.

• Distributional similarity using simple $n$-gram contexts has a number of advantages
  • Comparable precision to more NLP-rich techniques.
  • Better coverage.
  • Applicable “out of the box” to very large and noisy data sets.

• Resources available to broader community
  • Paraphrase evaluation data set publicly available
  • Mavuno will be released soon under an open source license
QUESTIONS?