Transformation-Based Error-Driven Learning

Brendan Walters
Comp 875
09/24/2009
Problem Domain: Text Tagging

- What is text tagging?
  - Some sort of markup, enabling understanding of language.
  - Can be word tags:
    - He ***will race/VERB*** the car.
    - He ***will not race/VERB*** the truck.
    - When will the ***race/NOUN*** end?
  - Or a tree structure:
Can have a very complete parse tree, tagging in multiple ways and possibly including semantic breakdowns of individual words:
Why do we care?

• Sometimes, meaning changes a lot
  – Transcribed speech lacks clear punctuation:
    “I called, John and Mary are there.”
    → I called John and Mary are there.
      • (I called John) and (Mary are there.) ??
      • I called ((John and Mary) are there.)
  – We can tell, but can a computer?
    • Here, needs to know about verb forms and collections
  – Can be important!
    • Quick! Wrap the bandage on the table around her leg!
    Imagine a robotic medical assistant with this one . . .
Where is this used?

• Any natural language task!
  – Translators: word-by-word translation does not always work, sentences need re-arranging.
  – It can help with OCR or voice transcription
    • “I need to writer. I'm a good write her.”
    • “to writer”? “a good write”?
    • → “I need to write her. I'm a good writer.”
  – Analysis of new languages
    • We've learned a bunch of words by point-and-ask. Now how the heck does this funny grammar work?!
• Human interaction – dialog structure.
  – “We should invite Julie to the party.” “Whose?” “Jim's surprise birthday party, remember?” “Oh right, we need to email her!”
Some Terms

- **Corpus**
  - Big body of text, annotated (expert-tagged) or not

- **Dictionary**
  - List of known words, and all possible parts of speech

- **Lexical/Morphological vs. Contextual**
  - Is it a word property (spelling) or surroundings (neighboring parts of speech)?

- **Semantics vs Syntax**
  - Meaning (definition) vs. Structure (phrases, parsing)

- **Tokenizer**
  - Separates text into words or other sized blocks (idioms, phrases . . . )

- **Disambiguator**
  - Extra pass to reduce possible tags to a single one.
English Part-Of-Speech Tagging

- Large available tagged corpora
  - Brown Corpus: 1 million words
  - Penn Treebank: 4.5 million words as of 1992
  - BNC: 100 million words (tagged by CLAWS4)
    - Accuracy appears to nearly match expert agreement
- Some ambiguity in the best case:
  - 3.5% expert annotator disagreement (Penn Treebank)
- Each has its own tagset, some are multi-tagged
  - Some are designed to link well with additional annotation (e.g., syntactic structure in Penn Treebank)
Some problems we face

• Classification challenges:
  – Large number of classes:
    • English POS: varying tagsets, 48 to 195 tags
    • Hebrew morphology: up to 300,000; ~1,934 in practice
  – Often ambiguous, varying with use/context
    • POS: There must be a way to go there; I know a person from there – see that guy there?
      (pron., adv., n., adj.)
  – Varying number of relevant features
    • Spelling, position, surrounding words, paragraph position, article topic . . .
Approaches: Rule-based vs. Stochastic

• Rule-Based
  – Usually some form of Constraint Grammar
  – Very high accuracy potential: EngCG2 (1997, Atro Voutilainen) achieves 99.5% (> expert agreement??)
  – Very large development cost:
    • Tokenizer: 8000+ built-in multiword expressions
    • Morphological analyzer: 90,000 entry lexicon
    • Contextual disambiguator: 3600 constraint regexps
  – Completely nonportable – specific to language and tags
  – Regular Expression matching not very interesting to us
Approaches: Rule-based vs. Stochastic

- **Stochastic**
  - Input corpora are used to train a tagger, often some form of Hidden Markov Model
  - Usually supervised (manually-tagged input corpora)
  - Often supplemented by manually-written rulesets
  - Generally require large hand-tagged training corpora (50k+ words)
  - Thus, most approaches are ill-suited to analysis of new or small-population languages.
  - But, a given method is potentially portable, especially to related languages or dialects.
An example: CLAWS

- Current version, CLAWS4, took 14 years to develop
  - Trained on ~500,000 word corpus from the BNC
  - 96-97% accuracy on BNC samples (58 tags for 100M words, 138 tags for 2M word Core Corpus)
- Components:
  - HMM tagger with 12,000 word dictionary
  - Manually-written “idiom lexicon” with 3,000+ rules
- Process:
  - Segment, tag with dictionary, correct for idioms, disambiguate with Hidden Markov Model
Why try a new approach?

- In early 90s, stochastic taggers were booming
  - The best methods outperformed all rule-based systems
- Originally conceived to determine whether rule-based systems could compete
- Statistical rule generation allowed testing of a large field of possible rulesets, although less fine-tuned than a complex grammar.
- Goals: small size, clarity, easy improvement, portability
  - HMMs are big, complex black boxes. Rules, you can read, understand, and correct, plus there are a lot fewer. Plus you can compare them!
Transformation-Based Learning

- Dictionary + rule set of sequentially-applied “patches”
- Learning process needs only an annotated corpus and set of rule templates
  - Output is smaller than an HMM and human-readable
    - Tagging process is visible at every step as well
    - Can evaluate function and identify paths to improvement
  - Can train on corpora 10x smaller than HMMs need
    - Excellent for use on relatively unstudied languages / tags
  - Has excellent unsupervised training capability
    - 96.8% on Penn Treebank with added unsupervised training
What does it do?

• **Transformation-Based Error-Driven Learning:**
  
  – First, a dictionary tags every word with its most common POS. So, “run” is tagged as a verb in both:
    
    “The **run** lasted 30 minutes” and “We **run** 3 miles every day”
  
  – Unknown capitalized words are assumed to be proper nouns, and remaining unknown words are assigned the most common tag for their three-letter ending.
    
    → “**blahblahous**” is probably an adjective.
  
  – Finally, the tags are updated by a set of “patches,” with the form “Change tag **a** to **b** if:”
    
    – The word is in context **C** (eg, the pattern of surrounding tags)
    
    – The word or one in a region **R** has lexical property **P** (eg, capitalization)
Eh?

- Patches are simple rules to edit the naive initial “most-likely” tags.
  - The can rusted.
  - Rule: Change (modal) to (noun) if: the preceding word is a (determiner).
    → The (determiner) can (modal verb) rusted (verb) . (.)
    → The (determiner) can (noun) rusted (verb) . (.)

- They often end up encoding “common sense knowledge” about the language.
  - Verbs don't come after “the” !!
So, do people actually use this?

- Initially, not as much as I think they should have.
  - Some enhancements were proposed, but a lot of focus was on English, where established tools existed.

- Now, quite a lot!
  - There seems to have been an explosion of papers in the last ~3-4 years; this coincides with an increase in interest in parsing other languages (a primary strength of this method), and doing comparative linguistics with computational tools.
    - Morphological analysis of Hebrew
    - Grapheme-to-phoneme prediction (~text to speech learning)
    - Bilingual (Chinese-English) expression extraction
    - Extraction of scrolling headlines in Turkish TV
What does it do?

- **Transformation-Based Error-Driven Learning:**
  - Initial state annotator (dictionary + lexical guess rule set) is trained from 90% of the corpus.
    - Initial results: error rate of 7.9% on Brown Corpus
  - Second pass (contextual patch rules) are trained from 5% of the corpus.
    - 71 patches reduced errors to 5.1%, a 35% improvement
  - Later versions added a lexical transformation patch set, and a second round of unsupervised learning (which improves performance dramatically and is very useful for languages for which there exist only tiny annotated corpora). We will address those later.
<table>
<thead>
<tr>
<th>#</th>
<th>From</th>
<th>To</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NN</td>
<td>VB</td>
<td>Previous tag is <em>TO</em></td>
</tr>
<tr>
<td>2</td>
<td>VBP</td>
<td>VB</td>
<td>One of the previous 3 tags is <em>MD</em></td>
</tr>
<tr>
<td>3</td>
<td>NN</td>
<td>VB</td>
<td>Previous tag is <em>MD</em></td>
</tr>
<tr>
<td>4</td>
<td>VBD</td>
<td>VBN</td>
<td>One of the previous 2 tags is <em>VBP</em></td>
</tr>
<tr>
<td>5</td>
<td>VBN</td>
<td>VBD</td>
<td>Previous tag is <em>NP</em></td>
</tr>
<tr>
<td>6</td>
<td>VBD</td>
<td>VBN</td>
<td>One of the previous 2 tags is <em>VBZ</em></td>
</tr>
<tr>
<td>7</td>
<td>VBN</td>
<td>VBD</td>
<td>Previous tag is <em>PP</em></td>
</tr>
<tr>
<td>8</td>
<td>POS</td>
<td>VBZ</td>
<td>Previous tag is <em>PP</em></td>
</tr>
<tr>
<td>9</td>
<td>VB</td>
<td>VBP</td>
<td>Previous tag is <em>NNS</em></td>
</tr>
<tr>
<td>10</td>
<td>VBP</td>
<td>VB</td>
<td>Previous tag is <em>TO</em></td>
</tr>
<tr>
<td>11</td>
<td>VB</td>
<td>VBP</td>
<td>Previous tag is <em>PP</em></td>
</tr>
<tr>
<td>12</td>
<td>JJ</td>
<td>NN</td>
<td>The surrounding tags are <em>DT</em> and <em>IN</em></td>
</tr>
<tr>
<td>13</td>
<td>VBD</td>
<td>VBN</td>
<td>Previous tag is <em>VBD</em></td>
</tr>
<tr>
<td>14</td>
<td>VB</td>
<td>NN</td>
<td>One of the previous two tags is <em>DT</em></td>
</tr>
<tr>
<td>15</td>
<td>IN</td>
<td>WDT</td>
<td>The surrounding tags are <em>NN</em> and <em>VBZ</em></td>
</tr>
<tr>
<td>16</td>
<td>NN</td>
<td>VBP</td>
<td>Previous tag is <em>PP</em></td>
</tr>
<tr>
<td>17</td>
<td>NP</td>
<td>NN</td>
<td>The surrounding tags are <em>START</em> and <em>NNS</em></td>
</tr>
<tr>
<td>18</td>
<td>NNS</td>
<td>NP</td>
<td>Following tag is <em>NP</em></td>
</tr>
<tr>
<td>19</td>
<td>RBR</td>
<td>JJR</td>
<td>One of the following 3 tags is <em>NNS</em></td>
</tr>
<tr>
<td>20</td>
<td>VBD</td>
<td>VBN</td>
<td>One of the previous 2 tags is <em>VB</em></td>
</tr>
</tbody>
</table>
Applying Patches, a similar example.

- Patches are simple rules to edit the naive initial “most-likely” tags.
  - It can rust.
    → It (pronoun) can (modal verb) rust (noun) . (.)
  - Rule #3: Change (noun) to (verb) if: the preceding word is a (modal).
    → It (pronoun) can (modal verb) rust (verb) . (.)

- This is a rule you might actually learn in a grammar class
  - Modal verbs (can, should, must, will . . .) are followed by verbs (or adverb-verb pairs, but NOT nouns).

- The importance of rules is determined by common errors from the dictionary, not necessarily grammatical importance.
How is it trained?

- **Transformation-Based Error-Driven Learning:**
  - “The tagger works by automatically recognizing and remedying its weaknesses, thereby incrementally improving its performance.”
  - (Brill 92)
How is it trained?

• Transformation-Based Error-Driven Learning:
  - A set of patch templates generate potential rules, and an greedy search procedure iteratively selects the best one.
  - This turns out to produce a provably optimal result for any given set of templates.

![Diagram of the training process](image-url)
How is it trained?

- Simple iterative greedy search over template results:
Patch Templates?

• Change tag $A$ to tag $B$ when:
  – The word 1 / 2 before or after has tag $W$
  – One of previous or next 2 / 3 words has tag $W$
  – The bracketing words are tagged $Z$ and $W$
  – The previous / next 2 words are tagged $Z$ and $W$
  – The current / previous word is / is not capitalized.
  – (After the 1993 thesis, other templates were added for the POS task, as well as different sets for different tasks / languages.)

• But these simple templates alone were remarkably effective at picking up all the context needed.
Performance: initial

- Initial State Annotator (ISA): dictionary + capitalization / word ending lexical guesser
  - 92.1% on Brown
- Contextual tagger:
  - 94.9% @ 71 patches
  - 3 neutral, 2 harmful
- What if we add template-based training to the ISA?
Lexical Patches

• What if we train the Initial State Annotator the same way that the contextual rules are trained?

• Templates: Change tag to $B$ / from $A$ to $B$ if:
  – the first/last 1-4 characters of the word are $X$.
  – deleting the length 1-4 pre/suffix results in a word.
  – adding the pre/suffix $X$ (len $\leq$ 4) results in a word.
  – the word $Y$ ever appears immediately before/after this.
  – the character $Z$ appears in the word.

• This has the neat side effect that the training reveals easily interpretable lexical trends in the sample. (More on this later!)
So, what do *these* patches look like?

<table>
<thead>
<tr>
<th>#</th>
<th>From</th>
<th>To</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>??</td>
<td>NNS</td>
<td>Suffix is <em>s</em></td>
</tr>
<tr>
<td>2</td>
<td>NN</td>
<td>NP</td>
<td>Can appear at the start of a sent.</td>
</tr>
<tr>
<td>3</td>
<td>??</td>
<td>VBN</td>
<td>Suffix is <em>ed</em></td>
</tr>
<tr>
<td>4</td>
<td>??</td>
<td>CD</td>
<td>Can appear to the right of $</td>
</tr>
<tr>
<td>5</td>
<td>??</td>
<td>VBG</td>
<td>Suffix is <em>ing</em></td>
</tr>
<tr>
<td>6</td>
<td>??</td>
<td>JJ</td>
<td>Character - appears in the word</td>
</tr>
<tr>
<td>7</td>
<td>??</td>
<td>JJ</td>
<td>Adding the suffix <em>ly</em> results in a word</td>
</tr>
<tr>
<td>8</td>
<td>??</td>
<td>RB</td>
<td>Suffix is <em>ly</em></td>
</tr>
<tr>
<td>9</td>
<td>??</td>
<td>NP</td>
<td>Can appear to the right of <em>Mr.</em></td>
</tr>
<tr>
<td>10</td>
<td>NN</td>
<td>VB</td>
<td>Can appear to the right of <em>will</em></td>
</tr>
<tr>
<td>11</td>
<td>NN</td>
<td>CD</td>
<td>Character 1 appears in the word</td>
</tr>
<tr>
<td>12</td>
<td>NN</td>
<td>JJ</td>
<td>Can appear to the right of <em>be</em></td>
</tr>
<tr>
<td>13</td>
<td>NN</td>
<td>NP</td>
<td>Character <em>S</em> appears in the word</td>
</tr>
<tr>
<td>14</td>
<td>??</td>
<td>NP</td>
<td>Character <em>M</em> appears in the word</td>
</tr>
<tr>
<td>15</td>
<td>VB</td>
<td>NN</td>
<td>Can appear to the right of <em>the</em></td>
</tr>
<tr>
<td>16</td>
<td>??</td>
<td>NP</td>
<td>Character <em>C</em> appears in the word</td>
</tr>
<tr>
<td>17</td>
<td>NNS</td>
<td>VBZ</td>
<td>Can appear to the right of <em>it</em></td>
</tr>
<tr>
<td>18</td>
<td>??</td>
<td>NP</td>
<td>Character <em>B</em> appears in the word</td>
</tr>
<tr>
<td>19</td>
<td>NN</td>
<td>NP</td>
<td>Character <em>A</em> appears in the word</td>
</tr>
<tr>
<td>20</td>
<td>??</td>
<td>NP</td>
<td>Prefix is <em>D</em></td>
</tr>
</tbody>
</table>

The first 20 lexical patches learned from the WSJ Corpus. (from Brill '93)
Interesting Results

- Because the output is human-interpretable, the resulting tagger can provide interesting insight into the content of the material on which it has been trained:

“A few differences between the transformations learned on the Wall Street Journal and those learned on the Brown Corpus are worth noting. First, the transformation indicating that a word to the right of a dollar sign is likely a number is a much more useful transformation in the Wall Street Journal (transformation number 4) than in the Brown Corpus (transformation number 19). Probably due to the fact that business tends to be male-dominated, the transformation found using the Brown Corpus that states that a word that can appear to the right of the word She is a past tense verb is not learned when trained on the Wall Street Journal.”

- (Eric Brill, 1993 thesis)
End of Lecture 1
So, how does this change things?

- The original method used almost the entire annotated Brown Corpus, 90% to train the dictionary and 5% to train the contextual patches. (~900k and ~50k words). Lexical guess rules (suffixes) were part of the dictionary.

- This enhanced ruleset generation offers the ability to dramatically reduce training requirements . . . BUT:
  - Generally, tagging words with their “most likely” (most observed) tag is about 93% accurate, while tagging the unknowns as singular nouns is about 19% accurate.
  - Issue: Less training data => smaller dictionary . . .
Smaller Dictionary Size == Ow.

- With a large corpus (~3M words), over 97% of words are recognized, but this drops off dramatically below ~500k words.

- @ ~22k training words, only around 81% of the test words are familiar – about 7 times as many unknowns
  - We need to improve our guessing!

![Graph showing the percentage of unseen tokens vs. size of training corpus.](image)
How do we compensate?

- We're using less of the corpus as annotated training data, but perhaps unsupervised training can help.
  - It's much easier to just get a big body of text than it is to manually annotate it!

- We can use the large unannotated corpus to collect statistics.

- Most lexical templates care about 'word-ness' and positioning, not known tags.
Huh?

• Learning without any annotation: Jabberwocky!

Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
All mimsy were the borogoves,
And the mome raths outgrabe.

"Beware the Jabberwock, my son!
The jaws that bite, the claws that catch!
Beware the Jubjub bird, and shun
The frumious Bandersnatch!"

He took his vorpal sword in hand:
Long time the manxome foe he sought --
So rested he by the Tumtum tree,
And stood awhile in thought.

And, as in uffish thought he stood,
The Jabberwock, with eyes of flame,
Came whiffling through the tulgey wood,
And burbled as it came!

One, two! One, two! And through and through
The vorpal blade went snicker-snack!
He left it dead, and with its head
He went galumphing back.

"And, has thou slain the Jabberwock?
Come to my arms, my beamish boy!
O frabjous day! Callooh! Callay!"
He chortled in his joy.

– Lewis Carroll – Jabberwocky. (From Through the Looking-Glass and What Alice Found There, 1872)
Learning without any annotation: Jabberwocky!

Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
All mimsy were the borogoves,
And the mome raths outgrabe.

"Beware the Jabberwock, my son!
The jaws that bite, the claws that catch!
Beware the Jubjub bird, and shun
The frumious Bandersnatch!"

He took his vorpal sword in hand:
Long time the manxome foe he sought
So rested he by the Tumtum tree,
And stood awhile in thought.

And, as in uffish thought he stood,
The Jabberwock, with eyes of flame,
Came whiffling through the tulgey wood,
And burbled as it came!

One, two! One, two! And through and through
The vorpal blade went snicker-snack!
He left it dead, and with its head
He went galumphing back.

"And, has thou slain the Jabberwock?
Come to my arms, my beamish boy!
O frabjous day! Callooh! Callay!"
He chortled in his joy.

- Determinant (/ Preposition / Possessive Pronoun) – *foo* – noun →
  - Adjective!
- Why not “the slithy toves”?
  - We don't know toves
- But what if we later saw “brilligious”?
  - [word - “-ous” = word] => Adj. (Which implies brillig = noun !)
Performance: Lexical Only

- Recall that using ~1M annotated words, the original tagger had 92% accuracy (lexical), improved to 95%.

- Results with 1,000 annotated sentences (~22k words, about 50 times less), compared against a successful non-Brill probabilistic method using a large suffix list:

<table>
<thead>
<tr>
<th>Method</th>
<th>Corpus</th>
<th>Unknown Words</th>
<th>Known Words</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexical Transformations</td>
<td>WSJ</td>
<td>77.5</td>
<td>93.6</td>
<td>90.5</td>
</tr>
<tr>
<td>Probabilistic Tagging</td>
<td>WSJ</td>
<td>71.7</td>
<td>95.4</td>
<td>91.0</td>
</tr>
<tr>
<td>Lexical Transformations</td>
<td>Brown - Penn Tags</td>
<td>74.7</td>
<td>93.3</td>
<td>89.9</td>
</tr>
<tr>
<td>Probabilistic Tagging</td>
<td>Brown - Penn Tags</td>
<td>71.6</td>
<td>94.7</td>
<td>90.3</td>
</tr>
<tr>
<td>Lexical Transformations</td>
<td>Brown - Orig Tags</td>
<td>71.0</td>
<td>92.5</td>
<td>88.4</td>
</tr>
<tr>
<td>Probabilistic Tagging</td>
<td>Brown - Orig Tags</td>
<td>65.1</td>
<td>94.8</td>
<td>89.1</td>
</tr>
<tr>
<td>Lexical Transformations</td>
<td>Old English</td>
<td>67.2</td>
<td>88.6</td>
<td>84.2</td>
</tr>
</tbody>
</table>
Performance: Combined, WSJ

- Adding the contextual transformations gives a boost of about 2% again, lessening as the base improves.
- Larger training sets help, and performance is close to the old method at ~10% of the training set size.

<table>
<thead>
<tr>
<th>Training Size</th>
<th>Unknown Word Accuracy</th>
<th>Known Word Accuracy</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000 Sentence Lexical and Contextual Training Corpora</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexical Transformations</td>
<td>77.5</td>
<td>93.6</td>
<td>90.5</td>
</tr>
<tr>
<td>Lexical and Contextual Transformations</td>
<td>81.2</td>
<td>95.3</td>
<td>92.7</td>
</tr>
<tr>
<td>Probabilistic (Small Suffix List)</td>
<td>70.4</td>
<td>95.4</td>
<td>90.7</td>
</tr>
<tr>
<td>Probabilistic (Big Suffix List)</td>
<td>71.7</td>
<td>95.4</td>
<td>91.0</td>
</tr>
<tr>
<td>4,000 Sentence Lexical and Contextual Training Corpora</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexical Transformations</td>
<td>81.3</td>
<td>93.4</td>
<td>92.2</td>
</tr>
<tr>
<td>Lexical and Contextual Transformations</td>
<td>84.3</td>
<td>95.5</td>
<td>94.4</td>
</tr>
<tr>
<td>Probabilistic (Big Suffix List)</td>
<td>75.2</td>
<td>96.1</td>
<td>94.1</td>
</tr>
</tbody>
</table>
Performance: Brown & WSJ+

- Complete results on the Brown Treebank:

<table>
<thead>
<tr>
<th>Tag Set</th>
<th>Unknown Word Accuracy</th>
<th>Known Word Accuracy</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexical Transformations</td>
<td>Penn</td>
<td>74.7</td>
<td>93.3</td>
</tr>
<tr>
<td>Lexical and Contextual</td>
<td>Penn</td>
<td>80.8</td>
<td>94.5</td>
</tr>
<tr>
<td>Statistical Tagging</td>
<td>Penn</td>
<td>71.6</td>
<td>94.7</td>
</tr>
<tr>
<td>Lexical Transformations</td>
<td>Brown</td>
<td>71.0</td>
<td>92.5</td>
</tr>
<tr>
<td>Lexical and Contextual</td>
<td>Brown</td>
<td>75.0</td>
<td>94.6</td>
</tr>
<tr>
<td>Statistical Tagging</td>
<td>Brown</td>
<td>65.1</td>
<td>94.8</td>
</tr>
</tbody>
</table>

- Adding 50,000 sentences (~1M words) gives superior results to the old method (WSJ corpus):

<table>
<thead>
<tr>
<th></th>
<th>Unknown Word Accuracy</th>
<th>Known Word Accuracy</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexical and Contextual Transforms</td>
<td>83.4</td>
<td>95.7</td>
<td>95.3</td>
</tr>
<tr>
<td>Probabilistic</td>
<td>76.7</td>
<td>96.3</td>
<td>95.7</td>
</tr>
</tbody>
</table>
Extension: Unsupervised Learning

- What if we had no annotated text at all?
  - We need something: a general dictionary is easier to provide, and in fact a prerequisite to annotation.
  - We need new templates to handle ambiguous tags (all possible tags), and the lack of model text.

- New templates: Change tag from $X$ to $Y$ if:
  - The previous/next tag is $T$
  - The previous/next word is $W$

(Where $X$ is a set of 2+ tags, and $Y \in X$)

- This is a disambiguator only – it does not switch tags, only reducing them.
Using an Unsupervised Learning Rule

• Our old friend:
  – The can rusted.
    → The (determiner) can (modal verb / noun) rusted (verb) . (.)
  – Rule: Reduce (verb / modal verb / noun) to (noun) if: the preceding word is a (determiner).
    → The (determiner) can (noun) rusted (verb) . (.)
  – Alternate rule: Reduce (verb / modal verb / noun) to (noun) if: the preceding word is “the”

• So we get many of the same or similar rules.
Performance: Unsupervised TBEDL

- Keeping with the previous work, the dictionary is generated from a corpus in Brill's tests.
- This allows comparison with different sizes of dictionary as well as learning a bit about how well you can make this new form of dictionary from annotated text.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Training Corpus</th>
<th>Size (Words)</th>
<th>% Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Penn Treebank</td>
<td></td>
<td>120K</td>
<td>95.1</td>
</tr>
<tr>
<td>Brown Corpus</td>
<td></td>
<td>120K</td>
<td>95.6</td>
</tr>
<tr>
<td>Brown Corpus</td>
<td></td>
<td>350K</td>
<td>96.0</td>
</tr>
</tbody>
</table>
Semi-Supervised Learning

- What if we try the best of all worlds: use the unsupervised learner initially, then feed that to a supervised learner with a much smaller corpus than even before?
- The dictionary should help a lot, but many of those lovely ??-Tag rules won't do any good now.
- (We don't need to alter it for reduction, we use the disambiguated output.)
Semi-Supervised Learning

- Unannotated Text
  - Initial State
    - Annotated Text
      - Learner
        - Truth
          - Rules
            - Manually Tagged Text
              - Supervised Learner
                - Unsupervised Transformations
              - Unsupervised Learner
                - Initial-State Annotator: Unsupervised
                  - Unlabeled Text
Performance: Semi-Supervised

- Semi-supervised learning certainly helps, especially with small corpora (taking advantage of the really good starting dictionary).

- However, the benefit drops off as the corpus gets large enough to provide a good dictionary of the old style.

<table>
<thead>
<tr>
<th>Supervised Training Corpus Size (Words)</th>
<th>% Correct Using Unsupervised Transformations</th>
<th>% Correct Not Using Unsup. Transformations</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>95.1</td>
<td>90.8</td>
</tr>
<tr>
<td>400</td>
<td>95.4</td>
<td>91.8</td>
</tr>
<tr>
<td>1200</td>
<td>95.5</td>
<td>92.9</td>
</tr>
<tr>
<td>4000</td>
<td>95.7</td>
<td>93.9</td>
</tr>
<tr>
<td>7600</td>
<td>95.8</td>
<td>94.6</td>
</tr>
<tr>
<td>10300</td>
<td>96.0</td>
<td>95.1</td>
</tr>
<tr>
<td>22300</td>
<td>96.3</td>
<td>95.5</td>
</tr>
<tr>
<td>44400</td>
<td>96.6</td>
<td>96.1</td>
</tr>
<tr>
<td>61400</td>
<td>96.7</td>
<td>96.3</td>
</tr>
<tr>
<td>88200</td>
<td>96.8</td>
<td>96.5</td>
</tr>
</tbody>
</table>
Other extensions

- We spend a lot of time re-trying large sets of rules
  - Naïve search: $O(ta_{gs^4})$ tests per patch found
  - Data-driven generation: $O(\text{errors} \times \text{templates}) / \text{patch}$
- Big speed-up after the first few patches!
  - $\sim 100$ tags, corpus $\sim 60k$ words, error rates after first pass $< 10$
  - $\rightarrow$ 3 orders of magnitude improvement!

- Ramshaw and Marcus, 1995:
  - What if we throw out really bad ones, then re-enable them after “enough time has passed”?
  - Another order of magnitude improvement, but we lose the provable optimality, and need to choose a good threshold for badness, and when to put it back.
Intelligent rule generation & ordering

- Naïve time is on the order of 30 million seconds
- All candidates = all rules relevant to errors
- Candidates > Threshold culls rules relevant to too few: 3.5-4x boost
- Observe that the ordered method (stop when the best possible gain is less than current best) tests the full list nearly all of the time after 12 rules.

Source: Brendan Walters. CS187 project, Harvard University 2006. (Unpublished)
Other extensions

  - What if we throw out really bad ones, then re-enable them after “enough time has passed”?
    - Heuristic / calibrated choices are needed.
    - Also, this fails to offer much help past ~12 rules, in my analysis – you end up with well below 10x
  - Instead, I banned *all* tested inferior rules, but only until they can possibly help (record improvement > current best improvement + changes since ban).
    - Works as well initially, and gets better: 7.14x speedup over first 60 patches, 10.85 over next 60
    - Retains provable optimality
    - Can be sped up with a multiplier: 5x doubled speed again throughout, and kept the set optimal in practice.
Ban Lists: my improvement over R&M

- Initially, it is building up the list – but it starts creating massive (~10x) savings after ~11 passes.
- These savings persist, and in fact stabilize as more rules develop.
- Multiplying the number of changes before re-introduction is a cheap approximation to counting only relevant changes, so it is generally safe at 5x (+?); different introduction times make some passes longer, but on average, we get a stable 2x gain.

Source: Brendan Walters. CS187 project, Harvard University 2006. (Unpublished)
What's happening with this today?

- Increasing numbers of customized extensions and novel uses, often specifically implemented as a Brill-style tagger. As mentioned before:
  - There seems to have been an explosion of papers in the last ~3-4 years; this coincides with an increase in interest in parsing other languages (a primary strength of this method), and doing comparative linguistics with computational tools.
  - Morphological analysis of Hebrew
  - Grapheme-to-phoneme prediction (~text to speech learning)
  - Bilingual (Chinese-English) expression extraction
  - Extraction of scrolling headlines in Turkish TV
  - See the final “Interesting Further Reading” slide for these and more.

- And other forms of error-driven training have sometimes drawn from this work in principle.
References

- **Reading List**
    - http://acl.ldc.upenn.edu/W/W95/w95-0101.pdf
    - http://acl.ldc.upenn.edu/W/W95/W95-0107.pdf

- **Primary Paper (not in reading list, 154 pages)**
    - http://repository.upenn.edu/ircs_reports/191/
References

  • http://portal.acm.org/citation.cfm?id=974526

  • http://portal.acm.org/citation.cfm?id=218367

  • http://ucrel.lancs.ac.uk/papers/ClawsWordTaggingSystemRG87.pdf
  • http://ucrel.lancs.ac.uk/claws/


- The Penn Treebank Project & The Brown Corpus Manual
  • http://www.cis.upenn.edu/~treebank/ & http://icame.uib.no/brown/bcm.html
Interesting Further Reading


  - http://acl.ldc.upenn.edu/P/P06/P06-1084.pdf

  - http://www.isca-speech.org/archive/interspeech_2006/i06_1742.html


  - http://linkinghub.elsevier.com/retrieve/pii/S0957417408002704


  - http://www.springerlink.com/index/h6w2374n21493957.pdf