Ling/CSE 472: Introduction to Computational Linguistics

4/16
Evaluation & Error Analysis

Overview

- Evaluation in computational linguistics
- Error analysis: One nice example
- Term project specs
- Reading questions

Why Evaluation?

- Good evaluation is essential to NLP research:
  - Verifies performance of process
  - Provides feedback on system changes
  - An essential part of the development process
  - Necessary for system comparisons
  - Provides information to potential users (and funders)

Ingredients

- Gold standard (“ground truth”)
- Evaluation metric: What you’ll count
- Baseline or baselines: What you’ll compare against
- Upper bound (optional)

Design considerations

- What system component is being evaluated? ex:
  - Parser
  - Language model
  - POS tagger
- What is the application? ex:
  - automated email response
  - travel dialogue system
  - document retrieval

Design considerations

- What are the evaluation criteria?
  - Accuracy
  - Coverage
  - Speed
  - Efficiency
  - Compatibility
  - Modifiability
  - Ease of use
  - Cost
  - …

Design considerations

- What is the goal of the evaluation?
    - Validation: Does the system do what you meant it to do?
    - Regression testing: Do recent changes improve performance, and/or lose any coverage?
    - Intrinsic evaluation: How well does it perform the specific task?
    - Extrinsic evaluation: How does it impact overall system performance?
    - Hypothesis testing: Can X information be used to aid in Y task?

Design considerations

- What resources are available?
  - Annotated corpora (e.g., Treebanks, aligned corpora)
  - Specialized corpora from application domain
  - Dictionaries and lexicons (e.g., pronunciation dictionaries, WordNet)
  - Test suites
    - Systematic collections of acceptable and unacceptable examples of specific phenomena
    - Generally hand built for each system and evaluation
    - Efforts to create shared resources, e.g. TSNLP (English, French, German)
- Are there standard corpora or evaluation metrics for the task?

Data for evaluation

- Separate test data from training and development data
- Use standard data sets where possible, to facilitate replication of results and inter-system comparison
  - Data often the result of challenges or shared tasks sponsored by NIST or various workshops
  - Data often distributed through LDC or ELRA, or more recently Kaggle
- Where there is no standard, clearly define the data and make it available to others

Handling data: Machine learning paradigm

- Divide data into training, development and test sets:
  - Training: Original input to stochastic model
  - Development: “Pretest” for tuning parameters (to avoid over-fitting on training data)
  - Test: Held-out data to measure generalizability of the system
- Dev and test data are always annotated (“gold standard”)
- Training data may be annotated (supervised learning) or not

Handling data: Knowledge engineering/rule-based paradigm

- “Training” data is examined by developer for rule development
- Training data is also used for regression testing
    - Does the current system analyze the same items as the previous one did?
    - Does the current system assign the same analyses as the previous one did?
- Test data is ideally unseen by both the system and the developer

Handling data: Knowledge engineering/rule-based paradigm

- Dealing with out-of-vocabulary words:
    - Measure overall performance anyway
    - Select only test data with known vocabulary
    - Add lexical entries for unknown words and test remaining system
- Error analysis can be very informative

Evaluation metrics

- Quantifiable measures
- Human inspection may be best, but can be impractical
- Automated approximations are cheaper, and especially valuable during system development
- The best metrics are those aligned with the goals of the application
- Use standardized metrics where available
- If none are available, clearly define the metrics used and use more than one

Example Metric: Precision and Recall

- Originally developed (and named) for Information Retrieval as a metric for search effectiveness

- Extended to the evaluation of various NLP tasks, especially ones involving categorization/labeling

- Provides measures of how correct (precision) and how thorough (recall); these goals are usually in tension

Precision and Recall

- Precision:
  - Proportion of results of the system that were correct

$$P = \frac{\#\text{correct results}}{\#\text{results returned}}$$

- Recall:
  - Proportion of correct results that were returned by system

$$R = \frac{\#\text{correct results}}{\#\text{results in gold standard}}$$

F-measure (combination of P and R)

$$F = \frac{(\alpha + 1) \times P \times R}{\alpha P + R}$$

- Varying the constant α affects the weight of Precision vs. Recall; increasing α increases the weight of Recall in the measure

- If α =1, Precision and Recall are equally weighted:

$$F = \frac{2 \times P \times R}{P + R}$$

Precision and Recall: Questions

- Why do we need to measure both precision and recall?
- Why would precision and recall be in competition?
- What is an example of an application that favors high recall?
- What is an example of an application that favors high precision?

Example Metric: BLEU score

- Automatic evaluation metric for machine translation (MT) (Papineni et al, ACL 2002)
- Measures similarity between system output and reference translations (gold standard)
- Measures lexical choice (unigrams), fluency (n-grams), and something like syntax (n-grams)
- Weighted average of the number of n-gram overlaps with reference translations: Weighted geometric mean of unigram, bigram, trigram and 4-gram scores

BLEU score

- Useful for comparing MT systems and tracking systems over time
- No meaningful units; for comparison, data sets must be the same
- One of several automatic MT evaluation metrics useful for development feedback
- Oft criticized
- Best MT evaluations use human raters (fluency, adequacy, edit distance)

Example metric: Parseval

- Automatic metric for evaluating parse accuracy when an annotated corpus is available
- Compares parser output to reference parses (gold standard)
- Evaluates component pieces of a parse
- Does not require an exact match: gives credit for partially correct parses

Parseval measures

- Labeled precision:

$$\frac{\text{\# of correct constituents in candidate parse}}{\text{total \# of constituents in candidate parse}}$$

- Labeled recall:

$$\frac{\text{\# of correct constituents in candidate parse}}{\text{total \# of constituents in gold standard parse}}$$

  - Constituents defined by starting point, ending point, and non-terminal symbol of spanning node

- Cross brackets: average number of constituents where the phrase boundaries of the gold standard and the candidate parse overlap

  - Example overlap: ((A B) C) v. (A (B C))

Issues with Parseval

- Parseval is the standard metric. However:
- Flawed measure:
  - Not very discriminating -- can do quite well while ignoring lexical content altogether
  - Sensitive to different styles of phrase structure (does particularly well on the flat structure of the Penn Treebank)
  - Too lenient sometimes, too harsh at others
  - Single errors may be counted multiple times
- Relevant only for CFGs (Phrase Structure Grammars)
- Most important question is: How well does it correlate with task improvement? Not clear.

Comparison

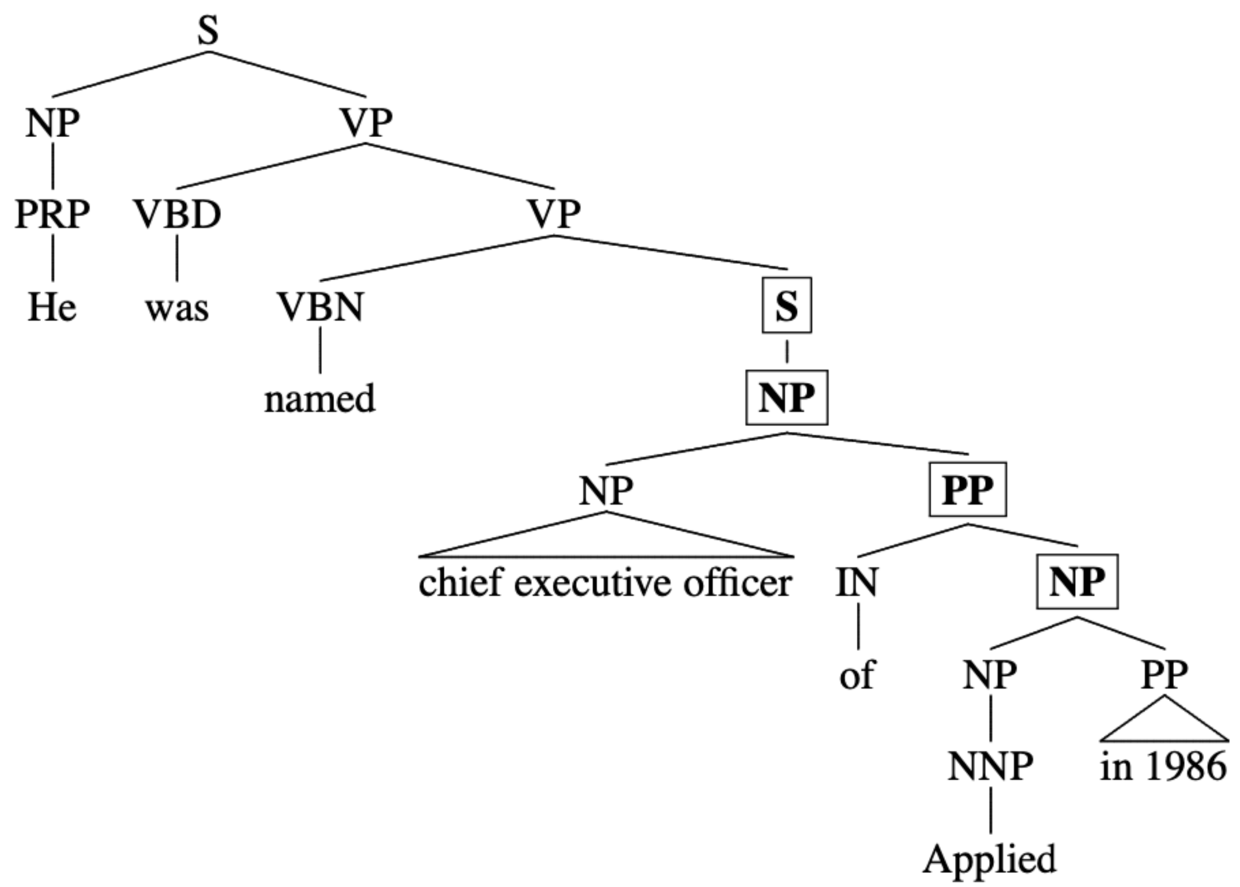
- Baseline: What you must beat
- Competing systems: What you want to beat
- Upper Bound (ceiling): What you aspire to
- Any difference must be statistically significant to count
- When comparing components, the rest of the system must be kept constant

Error analysis

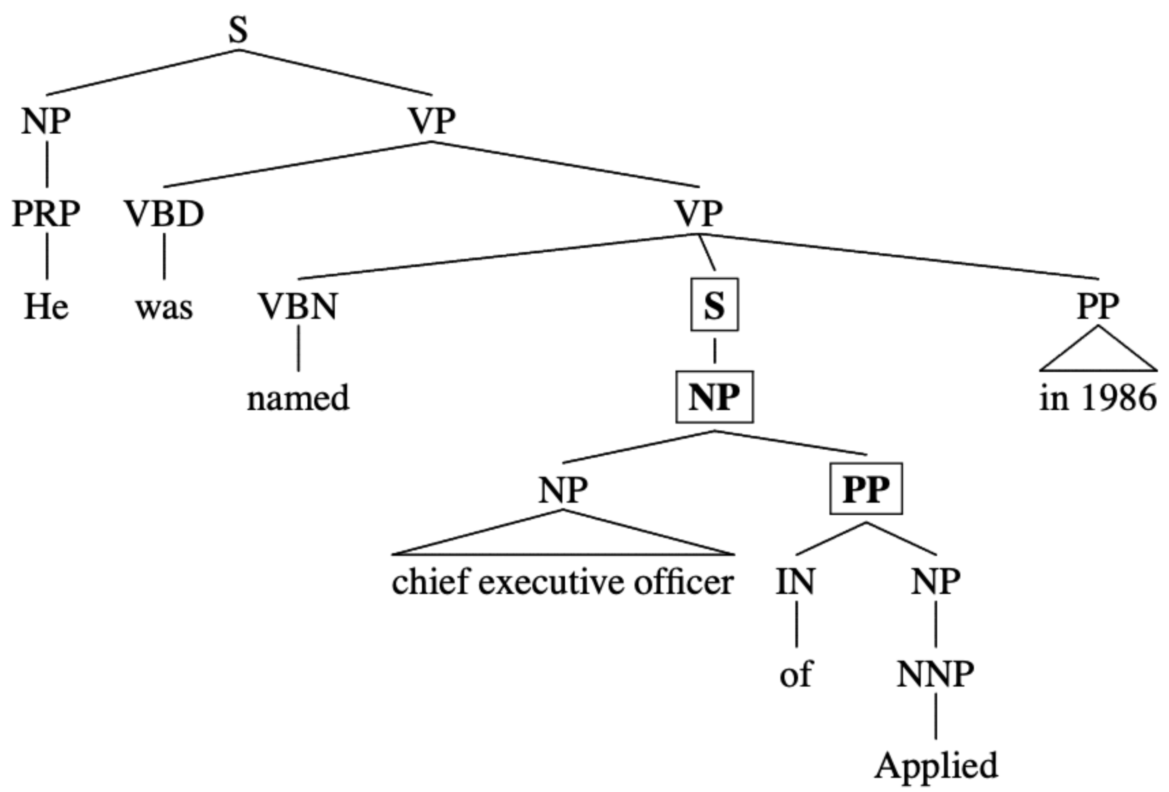
- What types of errors does the system make?
- What are the likely causes of each error type?
- How could the system be improved?
  - Which changes would have the most impact?
- How do the errors affect larger system performance?
- Note difference between error analysis and debugging

Kummerfeld et al 2012

- Target of evaluation: Constituent parsing of English
- Training data: Hand-labeled constituent trees (WSJ text, Penn Treebank)
- Output: Parse trees (CFG-style)
- Different parsers = different parse selection strategies
- Standard evaluation metric: Parseval



(a) Parser output



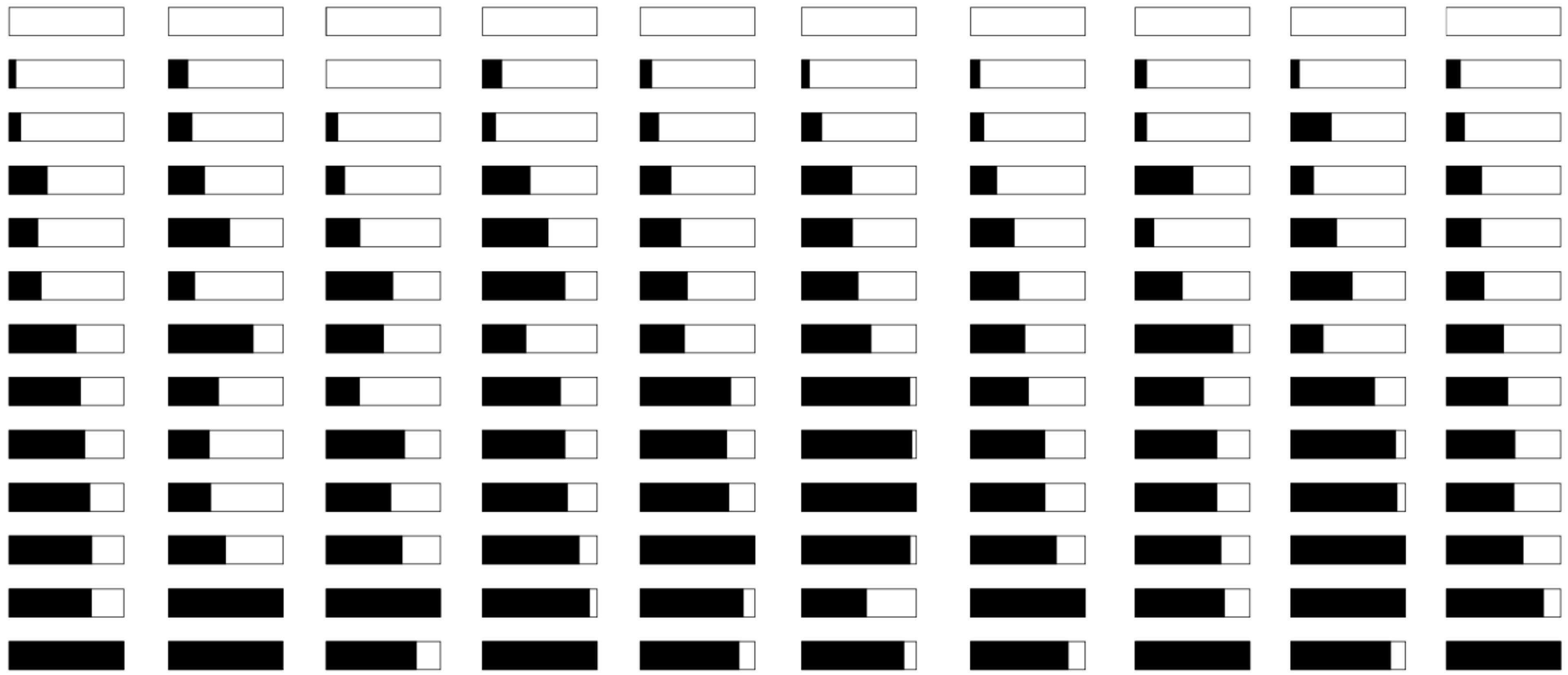
(b) Gold tree

Kummerfeld et al 2012

- Group individual errors at the constituent level into patterns representative of larger ‘mistakes’
- Look across those mistake types for better understanding of:
    - What is hard about constituent parsing of English
    - Strengths of different algorithms
    - Where the difficulties arise in cross-domain parsing

Error Type	Occurrences	Nodes Involved	Ratio
PP Attachment	846	1455	1.7
Single word phrase	490	490	1.0
Clause Attachment	385	913	2.4
Modifier Attachment	383	599	1.6
Different Label	377	754	2.0
Unary	347	349	1.0
NP Attachment	321	597	1.9
NP Internal Structure	299	352	1.2
Coordination	209	557	2.7
Unary Clause Label	185	200	1.1
VP Attachment	64	159	2.5
Parenthetical Attachment	31	74	2.4
Missing Parenthetical	12	17	1.4
Unclassified	655	734	1.1

Table 3: Breakdown of errors on section 23 for the Charniak parser with self-trained model and reranker. Errors are sorted by the number of times they occur. Ratio is the average number of node errors caused by each error we identify (i.e. Nodes Involved / Occurrences).



Parser	F-score	PP Attach	Clause Attach	Diff Label	Mod Attach	NP Attach	Co-ord	1-Word Span	Unary	NP Int.	Other
Best		0.60	0.38	0.31	0.25	0.25	0.23	0.20	0.14	0.14	0.50
Charniak-RS	92.07										
Charniak-R	91.41										
Charniak-S	91.02										
Berkeley	90.06										
Charniak	89.71										
SSN	89.42										
BUBS	88.63										
Bikel	88.16										
Collins-3	87.66										
Collins-2	87.62										
Collins-1	87.09										
Stanford-F	86.42										
Stanford-U	85.78										
Worst		1.12	0.61	0.51	0.39	0.45	0.40	0.42	0.27	0.27	1.13

Table 2: Average number of bracket errors per sentence due to the top ten error types. For instance, Stanford-U produces output that has, on average, 1.12 bracket errors per sentence that are due to PP attachment. The scale for each column is indicated by the Best and Worst values.

Kummerfeld et al 2012

- Group individual errors at the constituent level into patterns representative of larger ‘mistakes’
- Look across those mistake types for better understanding of:
    - What is hard about constituent parsing of English
    - Strengths of different algorithms
    - Where the difficulties arise in cross-domain parsing

Overview

- Evaluation in computational linguistics
- Error analysis: One nice example
- Term project specs
- Reading questions

Term project specifications

- find an NLP package described in a peer-reviewed paper which reports results using a quantitative evaluation metric, such as precision/recall;
- run it on the dataset that comes with it (or, in some exceptional cases, on a different dataset),
- and perform careful error analysis over the results (for ~100 errors + some number of correct outputs for comparison)

Term project specifications

- All papers (packages) must include quantitative evaluation (e.g. in terms of precision and recall).
- Find a package that you can get running easily and quickly.
- Perform an analysis of the specific test items that the system does not get right by categorizing the errors, counting how many fall into which category, and finally providing a meaningful discussion about them, including either hypothetical or directly observed reasons for why the system is making these errors, as well as what implications this behavior might have.
- Your error analysis: its method, categories, and results, will be described in detail in the term paper.
- The term paper quality must be representative of the work that you did in this upper division course.

Reading questions

- How are baseline and upper bounds different from each other?
- What is the point of inter-annotator agreement when ultimately, we rely on the statistics of human annotators?
- Is automatic evaluation improving to the point where manual evaluation is not necessary anymore? Or are both still equally valuable for the foreseeable future. I guess this can extend to ML in general, beyond NLP.

Reading questions

- It is interesting to see that evaluation and testing are the most important part of production cycle. I wonder which evaluation types have a greater weight with respect to others, or it is purely arbitrary depending on the context in which the NLP system is developed?

- I was curious how often are new metrics for evaluation proposed and what distinctions do researchers usually make when devising them to make sure that the test is qualitatively sound.

- Do language differences affect the effectiveness/thoroughness of evaluation method (For example, one method is effective in English, but not in Chinese)?

Reading questions

- What is the value in reporting an F-score rather than reporting the precision and recall directly? Is it just for convenience? To fully understand the F-score, it seems like one would have to know both the F-score and the precision-recall weighting, which would still require reporting two numbers, which wouldn't actually be any more convenient than reporting precision and recall separate in the first place.

- How does categorizing error types for parsers inform design changes if (as I understand it) they're pretty much self-trained?

Reading questions

- I was wondering while reading how precision and recall can be defined to capture better the goal of a particular NLP system. It seems like in the parser example, for instance, one might want to have better precision at different levels of the tree, irrespective of machine learning benchmarks, depending on the level of the tree one is primarily interested in (if in linguistics) and the level one most requires accuracy in to complete a task (if in industry). Would it be useful to make, e.g., precision at lower levels of the tree weigh more as a development tool?

Reading questions

- How do we train machines/parsers to understand the mistakes people make, while still understanding that they're mistakes?
- Referencing the very end of the Kummerfeld paper, I'm wondering if there is a way to reliably train an ML system on both formal, error-free speech, and error-prone speech at the same time? Is that too difficult, or would it be the case that with enough data, the system could figure out what is and is not an error?

Reading questions

- Many of the evaluation methods discussed in Resnik and Lin 2010 rely heavily on manual tagging/labeling and this makes me wonder if more sophisticated methods of error analysis have been used for word-sense disambiguation processing (for example, my recent blog post was about Errudite, an application that made NLP error analysis more scalable, efficient, and testable without as much need as manual labeling of errors. This one evaluated more question answering systems).

- It makes sense that for a lot of extrinsic evaluation, human assessment is the most useful. I wonder whether adversarial networks could apply for some problem types as at least rough estimate of human assessment?

Reading questions

- Resnik & Lin 2010 defines training data as "a data set where input items are paired with the desired outputs, often as the result of manual annotation." Before reading this, I assumed that all training data was always manually annotated. Is it common for a system to be trained on data whose correct outputs have been defined by another computer (which, to my understanding, means the data could potentially be inaccurate)? Can such a system be reliably trained if its success is essentially dependent on the success of other NLP systems?

Reading questions

- Noticing that the parser paper is "older" in terms of NLP years (still valuable and the existence of updated parsers reinforces this), I am curious at how parsers are being used for NN and whether the evaluation methods are still desired for modern models. BERT vectors are known for encoding deep linguistic structures, some of which have been reversed engineered to better understand BERT. It would be interesting to see the performance of these extracted trees and whether there are latent structures that it has discovered that linguistics has ignored/thought unserviceable.

Reading questions

- How do algorithms distinguish the difference between homonyms like chill (make cool or cooler) from chill (lose heat)?

- I am curious of how to evaluate WSD in language that do not have word boundary. In these language, is evaluating WSD the same as evaluating the performance of word segmentation, since word segmentation also needed to deal with ambiguity?

- I wonder how systems would collect data on slang usage/newer words. Specifically on Twitter, I see new slang words that I've never encountered before and there are many usages where people unintentionally use the word incorrectly. These words include: yeet, simp, moot, etc. How would WSD evaluation work here?

Reading questions

- What are some applications of cross-language information retrieval systems? Are there CLIRs that work from a single search language to many other languages?

- The text mentions that CLIR systems "are about as effective as monolingual IR systems - since the inherent redundancy in documents and queries [...] compensates for the poor translation quality." While redundancy in documents makes sense to me as an explanation for this, wouldn't requiring redundancy in queries for good results significantly impact the usefulness of the tool? It seems like modern search engines like Google do not require all that much redundancy in queries.

Reading questions

- After reading about parsing in the Kummerfield paper and its conclusion that "issues, such as ungrammatical language and unconventional use of words, pose a greater challenge", I'm curious to know if we can successfully train a parser to recognize and handle these issues in error-prone language?
- Coming from the Kummerfeld reading, how exactly would pruning negatively affect the accuracy of the parser? (Perhaps they cut off too early in the tree?)
- Comparing with WER and perplexity, which is better when analyzing language model for Machine Translation?