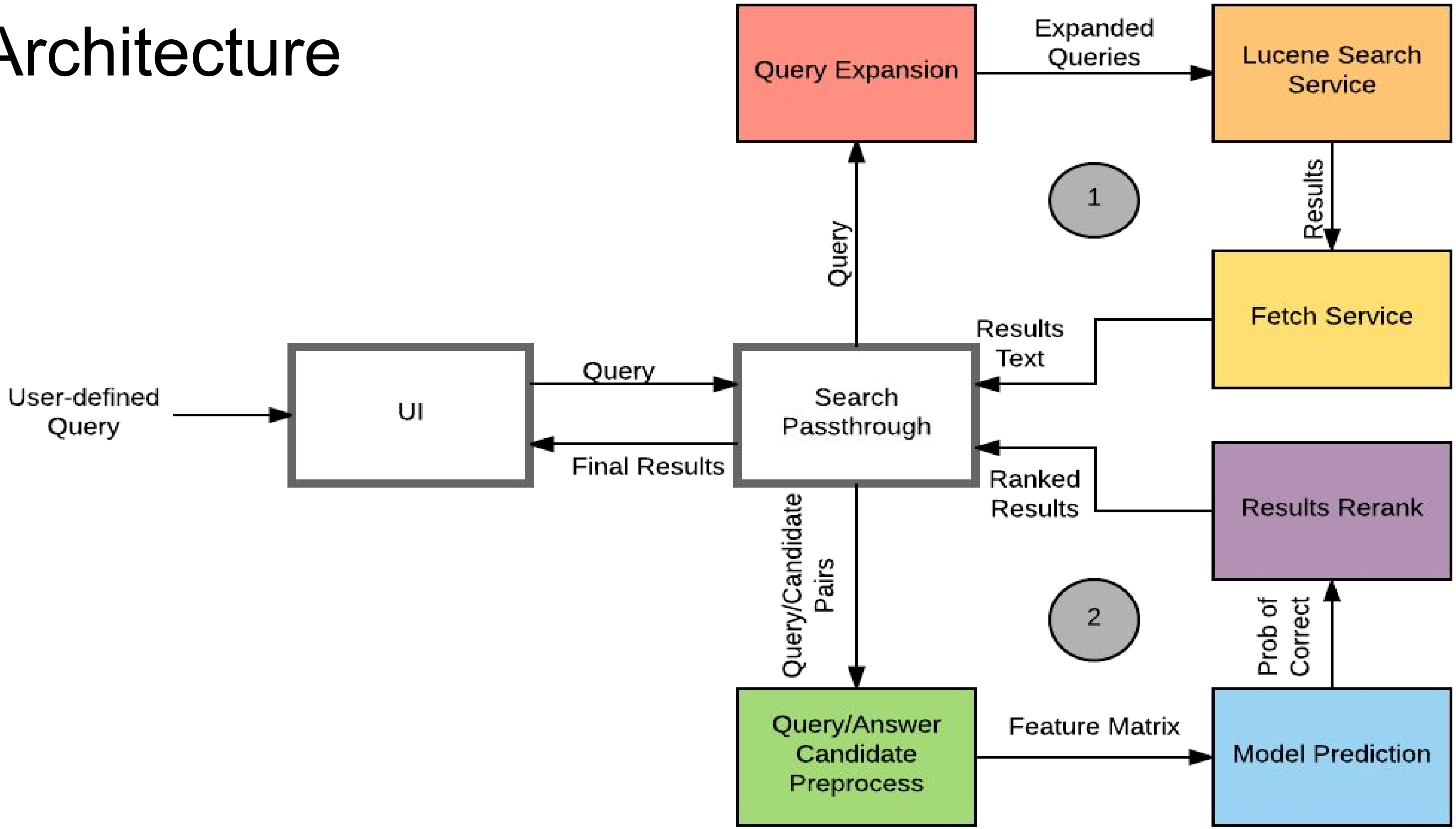
Total Recall: the New Google*

Rohan Tilva, Jordan Peykar, Matthew Lee, Bryan Ki, Hannah Cowley

* but bad this time

Architecture



Final K-Score Results

K	Baseline success @k
1	0.0
10	0.06
100	0.18
1000	0.30

K	Success @k with Query Expansion
1	0.04
10	0.16
100	0.48
1000	0.64

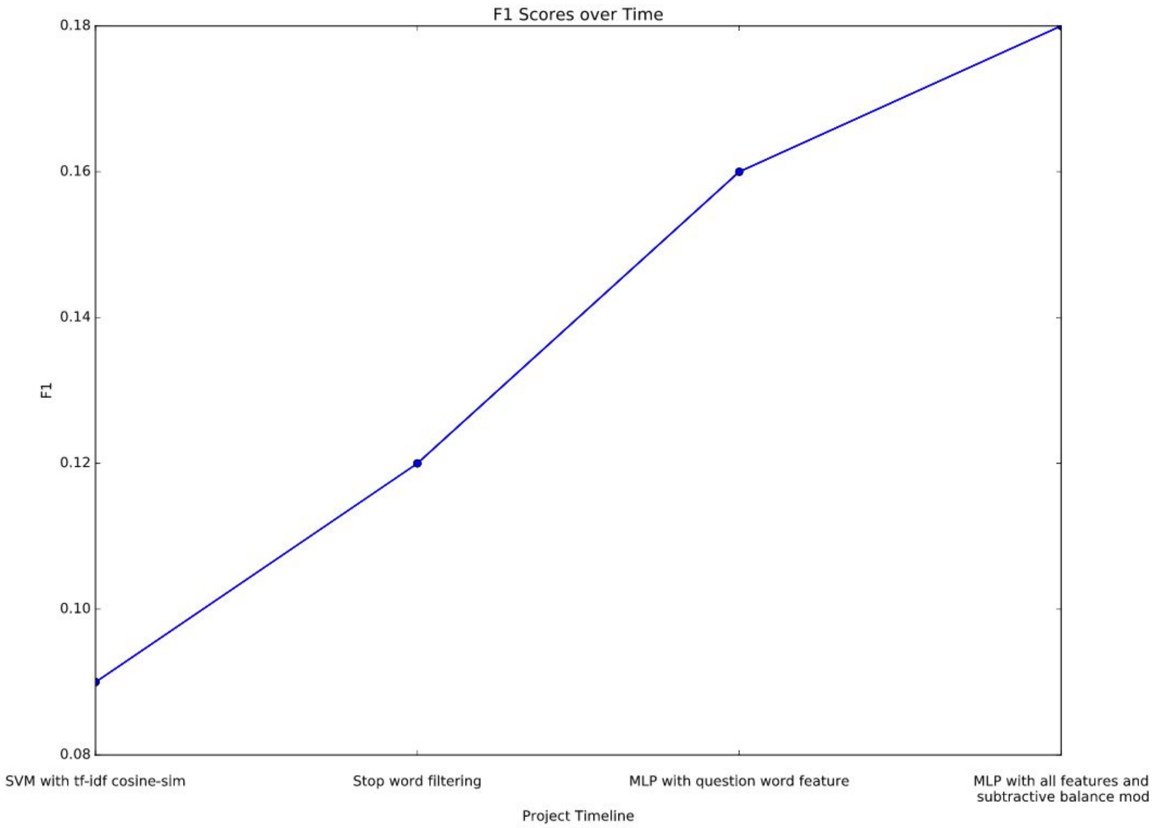
K	Final success @k (with MLP mini classifier)
1	0.0
10	0.0
100	.04
1000	.26

K	Final success @k (with logistic regression mini classifier)
1	0.0
10	.01
100	.09
1000	.30

Our MLP did not provide truly continuous probability estimations (they were skewed)… so we tried logistic regression.

F1 Results: MLP (with all features)

- Final Dev Results:
  - F1: 0.18
  - P: 0.11, R: 0.54
- Final Test Results:
  - F1: 0.16
  - P: 0.11, R: 0.30
- Compare to old Dev:
  - F1: .133
  - P: .08, R: .55



F1 Results: Logistic Regression (with only 3 simple features)

- Final Dev Results:
    - F1: .14
    - P: .10, R: .24
- Final Test Results:
    - F1: .14
    - P: .10, R: .23
- While this model does worse on F1 scores, k-scores show improvement over the MLP

Query Expansion (Learning to Paraphrase for Question Answering)

- The Good
  - (NLTK) WordNet
  - Stem each word
    - Synonyms for adverb, adjectives, and verbs only
    - Synonyms only found for non-common verbs
  - List of queries formed, search called on all
- The Bad
  - Synonyms too broad and varied
    - Ie. “run” → “chop-chop”
    - Possible fix: receive synonyms based on higher threshold of similarity

Feature Improvements #1

- TF-IDF Cosine-Similarity:
  - Added feature for cosine similarity
  - Problem: too many common words
    - Fix: stopword filtering of 200 most common words
      - Added for both question & answer
  - Cosine-similarity from sparse vectors
    - Thanks scikit-learn!

Feature Improvements #2

- Question Words:
  - ["", who", "what", "when", "where", "why", "how", "is", "whom"]
  - Index corresponding to question word for given question
    - New feature vector
  - Hope: certain answers correlated with certain question words
    - Ie. “When” always correlated with “<date>” in answer
    - “How deep is the ocean?” vs. “Why is the ocean deep?”
      - Most of the words are the same
      - Question word = key distinguisher!

Feature Improvements #3

- Sums of Word Embeddings: The Good
  - Sum individual word embeddings into a vector
    - Do for query and answer
  - Encaptures **similar** words
  - Words in query/answer don't necessarily have to be same exactly
- Sums of Word Embeddings: The Bad
  - Does not account for length of sentences
  - Sum of two different sentences could have a high cosine similarity

Feature Improvement #4

- Jaccard Similarity
  - Ie. Percentage overlap between query and answer
  - Effort to quantify similarity in query/answer
  - Not perfect → words need to be exactly the same

$$\boxed{\frac{||\ \text{intersection(question, answer)}\ ||}{||\ \text{union(question, answer)}\ ||}}$$

Feature Improvements #5

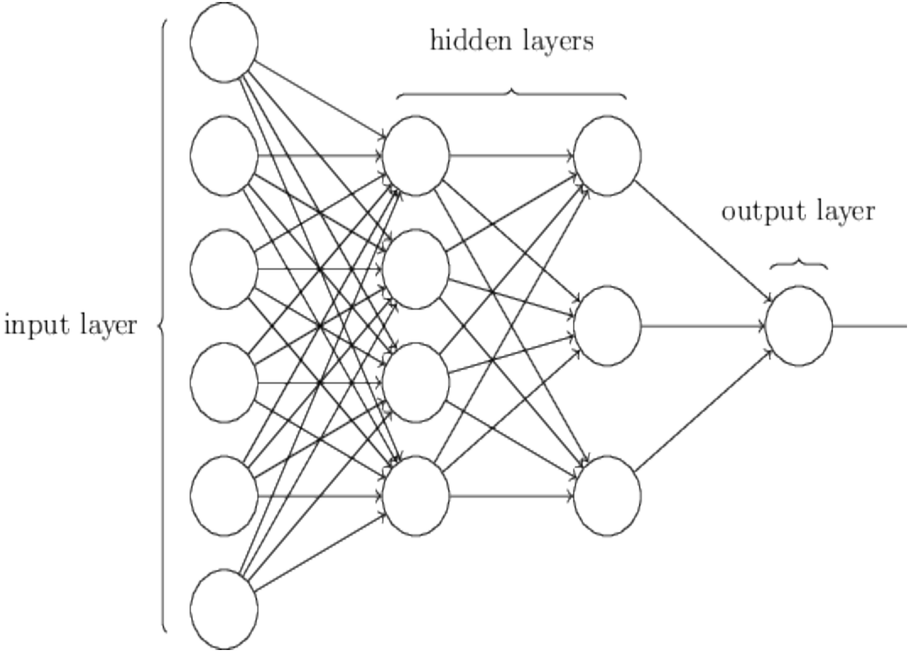
- Spacy Sentence Similarity Metric
  - The sentence similarity metric is able to use context clues to determine simila
    sentences
  - Word Embeddings usage
  - 4-layer CNN
    - Takes in context in 4-grams around the word in question to disambiguate
      the vocabulary

Feature Improvements #6

- Determinant of Word Embedding Vectors
    - (Boratto et al. 2016)
    - Used to measure linear independence (determinant) of word embedding vectors
    - F1 score had little change

Ranking and Classification

- Using a Multi-Layer Perceptron
  - Better F1 score than SVM
  - Hypothesis: data not linearly separable
  - So, MLP better!
  - Used MLPClassifier from scikit-learn
  - Fiddled with the following to maximize F1:
    - Parameters
    - Subtractive Balancing
    - Resampling



Ranking and Classification

- Using a Logistic Regression model (with only 3 features)
    - Better F1 score than SVM
    - Allowed for better probability distribution of whether sentence answers query
    - This allowed for better ranking, but didn’t have time to integrate this
- Integrated version still has the MLP (with full feature set)

Ranking and Classification

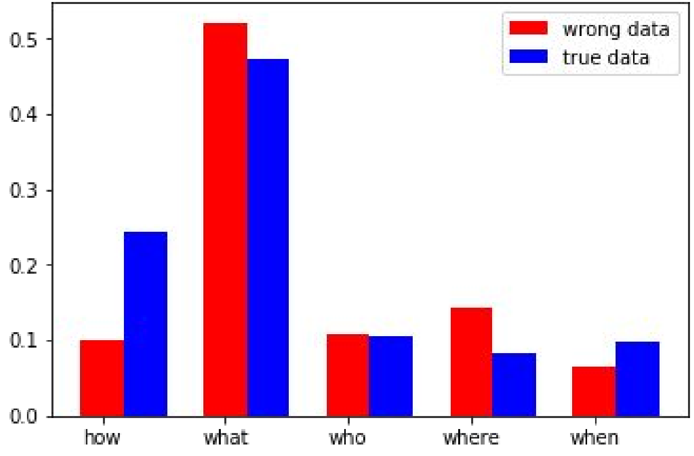
- Using Continuous Predictions for Ranking
  - Idea 1: Hard predictions (yes vs. no)
    - No method of determining which instances better answered question
  - Idea 2: Continuous predictions
    - Tuple of probabilities (one for “no”, one for “yes”)
    - Higher P(“yes”) → higher ranked

Ranking and Classification

- Modified Subtractive Balance
  - Idea 1: Train on all samples
    - Problem: number negative instances >> number positive instances
    - Skewed model to always predict “no”
  - Idea 2: Subtractive Balancing → number positive == number negative
    - Better F1 score
    - Problem: more negative instances in training data
  - Idea 3: Modified Subtractive Balancing → number negative = 1.8 * number pos
    - Best F1 score!

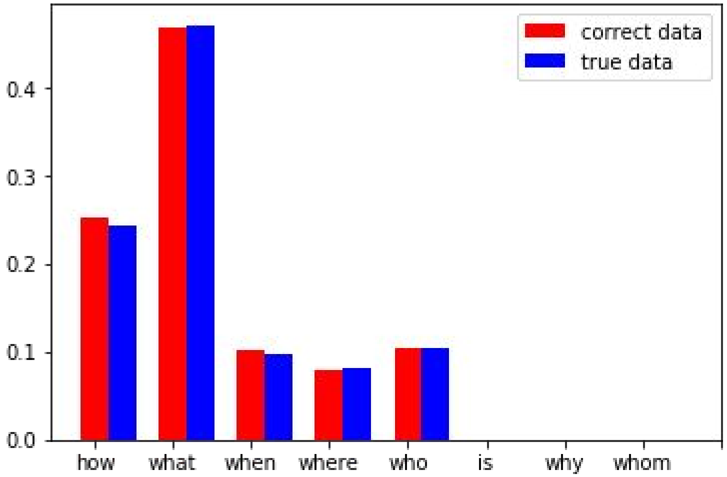
Error Analysis: Questions we got wrong (dev)

- Questions we got wrong
  - Ones that necessitated a numeric answer
    - How big is the Pacific Ocean? (Entity Recognition problem)
  - Where/What questions
- Shows that our precision is relatively low, but for a subset of question words in particular



Error Analysis: Questions we got right (on dev)

- Questions that were clear were classified correctly
  - Questions that had one clear answer
  - Questions that were highly specific
- A nice, even distribution of question words correctly classified relative to their corpus frequency!
  - Further confirmation: our system provides high recall, but low precision




NOT THE BEST
BUT STILL GOOD
MAR 19
1PCS
1/0.99
$0.99
NET WT.
TOTAL PRICE

Not the best, but still ok attempts

- Concrete-ly annotated POS tagging
  - Idea 1: Train model on POS tagging
    - Too slow
  - Conclusion:
    - We wish we could have incorporated this
    - Ie. matching subject in query to subject in answer could help a lot
    - Just need to try it on the quicker concrete fetch service (ran out of time)

Not the best, but still ok attempts

- Keras Sequential Model
  - Wanted to build a deep net
    - Problem: we don’t know ML
    - Didn’t know what input number of layers should be
    - Didn’t know the input shape
  - Back to MLP from scikit-learn

Areas for Improvement

- Increase F1 scores
    - Better feature engineering
    - Using a deep net instead
    - More positive data
        - Generate our own positive data?
    - MLP parameter adjustment
        - Not enough knowledge about ML to accurately change parameters
- Query Expansion
    - NLTK doesn’t have the best synonyms
    - Use other paraphrasing methods: SMT, PPDB

Citations

- L. Boratto, S. Carta, G. Fenu and R. Saia, "Exploiting a Determinant-Based Metric to Evaluate a Word-Embeddings Matrix of Items," 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), Barcelona, 2016, pp. 984-991.
- Chen, Tongfei and Van Durme, Benjamin. Discriminative Information Retrieval for Question Answering Sentence Selection. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 719-725. 2017.
- Dong, Li and Mallinson, Jonathan and Reddy, Siva, and Lapata, Mirella. Learning to Paraphrase for Question Answering. 2017.