Embrace the Noise

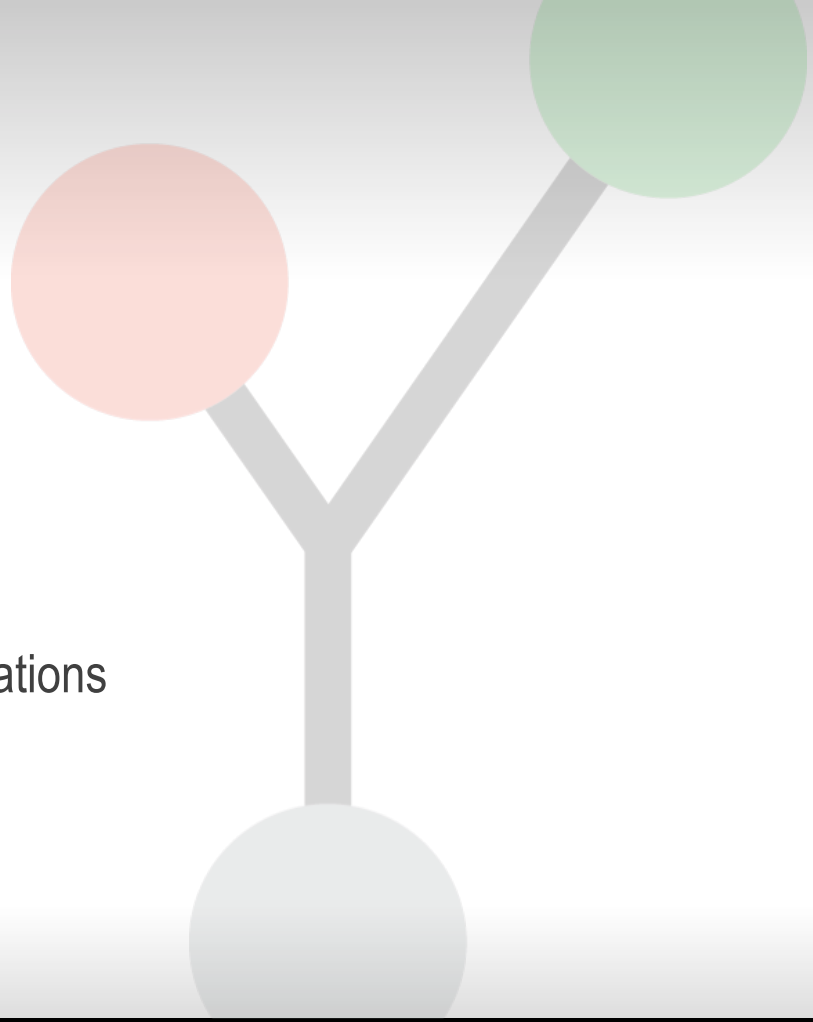
Seth Redmore

Lexalytics

@sredmore

Agenda

- ☐ Who is Lexalytics?
- ☐ Syntax, Semantics and Context
- ☐ Syntax parsing
- ☐ Ambiguities in syntax
- ☐ Applying matrices to resolve ambiguities
- ☐ Matrix Factorization for movie recommendations
- ☐ Matrix Factorization for syntax parsing
- ☐ Training the Matrix
- ☐ Results



Who is Lexalytics?

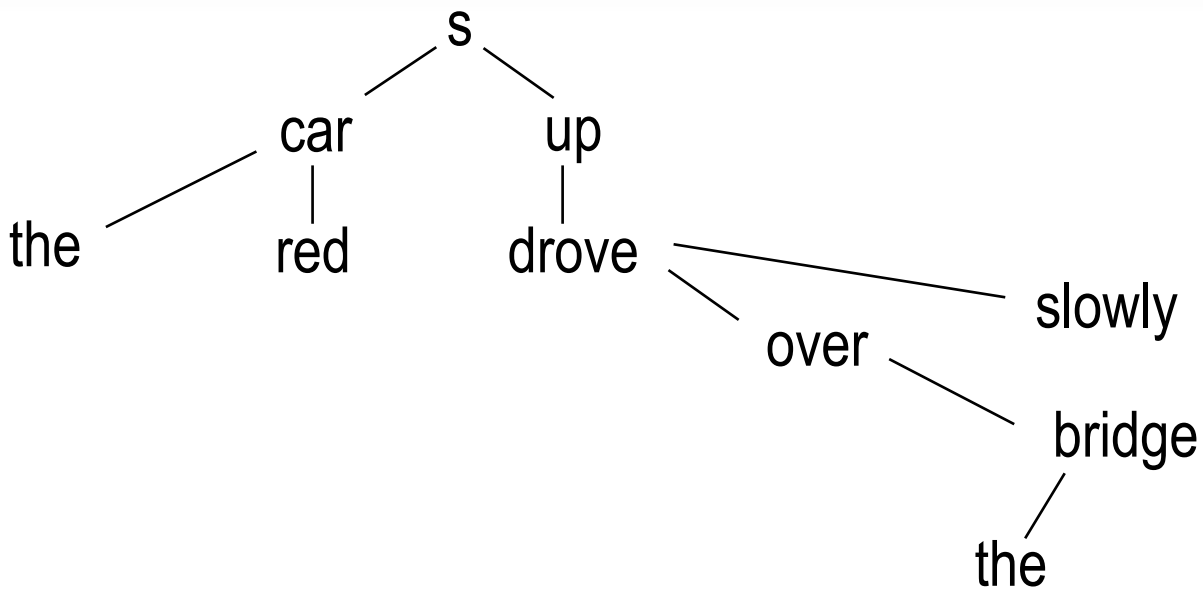
- Founded in 2003
- Text Analytics Engine
  - Entities, Sentiment, Topics, Intentions, Categories
- On-Premise, SaaS, Desktop
- Popular in Social Listening, Customer Experience Mgmt.
- Billions of documents/day processed across our customers

Syntax, Semantics, and Context

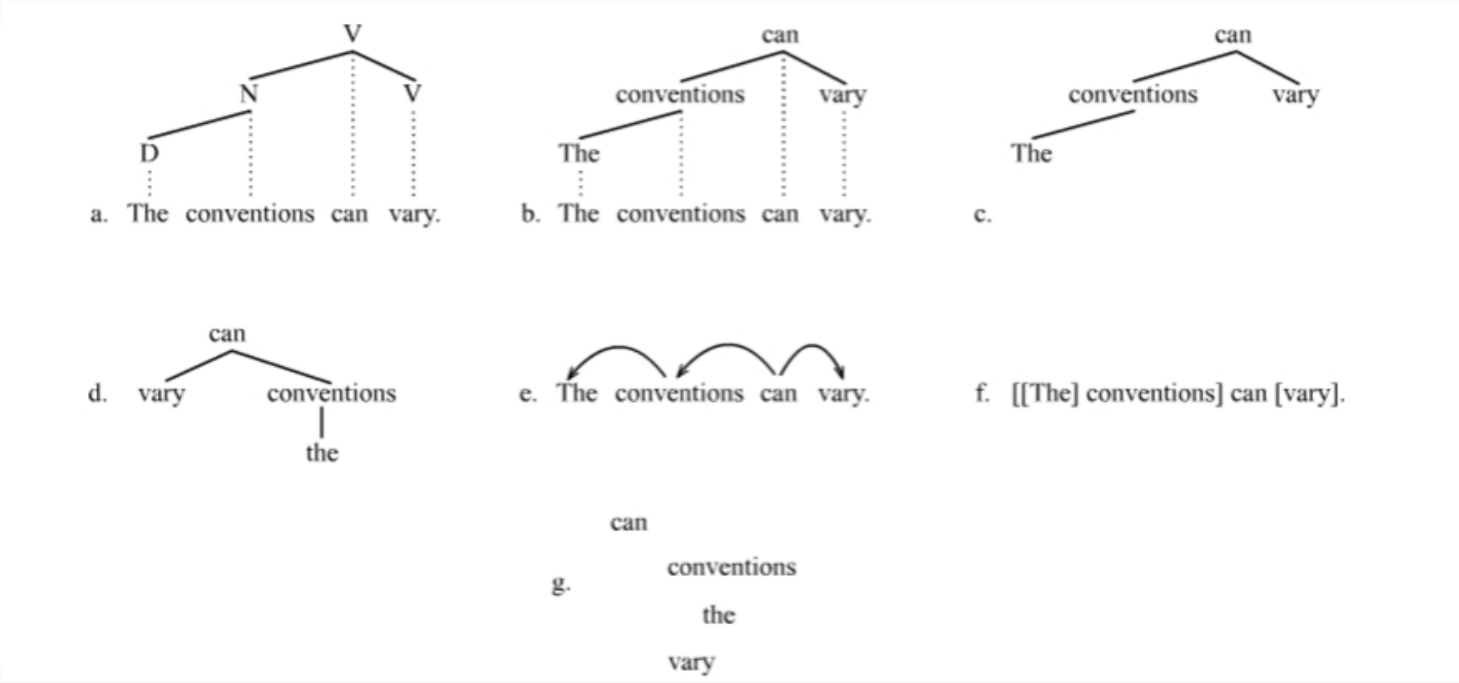
Each affect the “meaning” of the sentence.

- Syntax is the structure
- Semantics is meaning of individual words
- Context is fluffier, but could include
  - shared history
  - previous comments
  - physical location

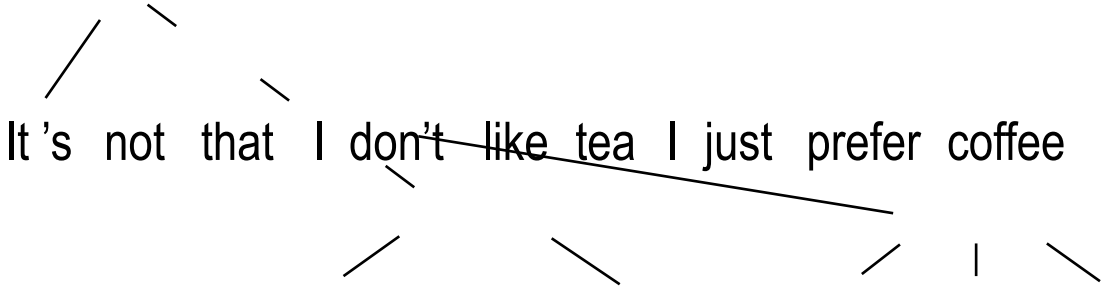
Let's talk syntax.



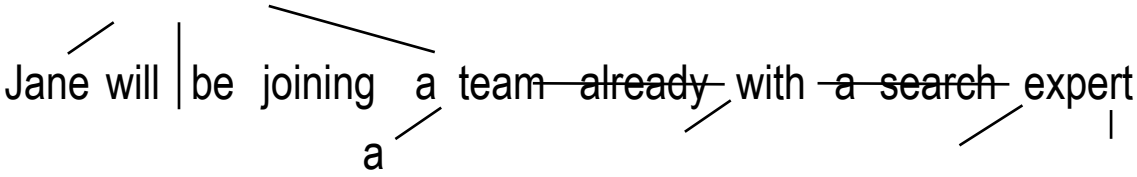
Dependency Parsing is Expensive…



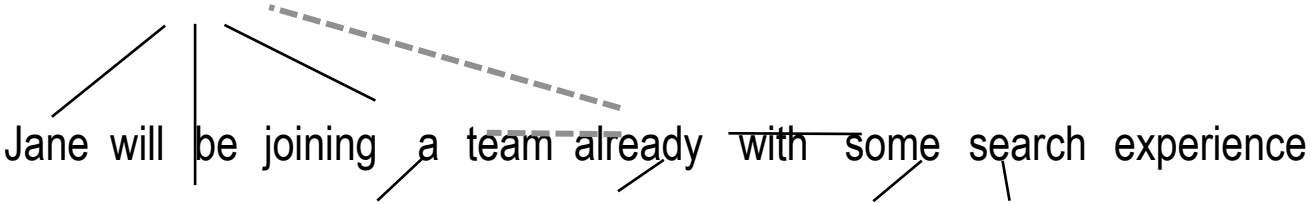
But, it’s important…



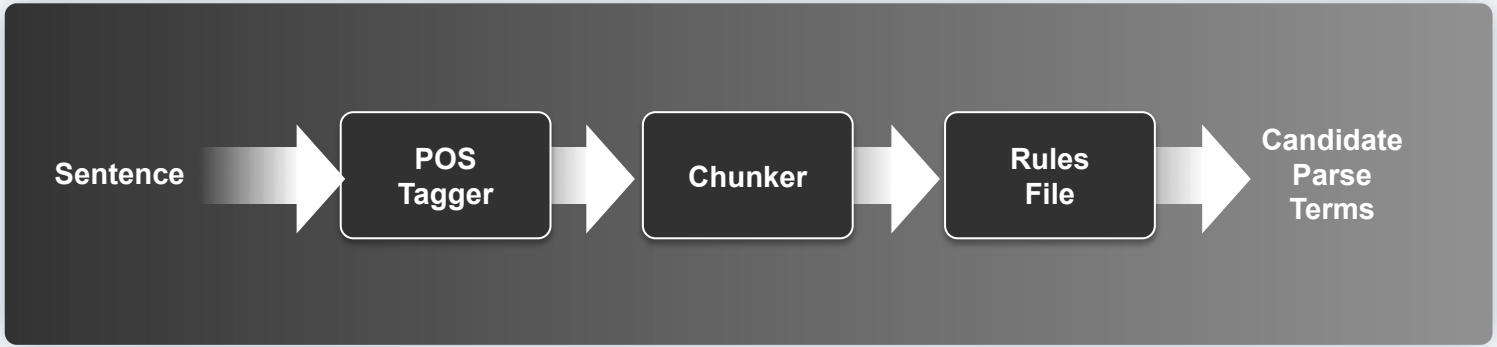
Meaning matters



Ambiguous syntax



Episode 4: A New Hope



Jane and her team

<Jane will be joining a team already with search experience>

- Pos Tag
  - <Jane_NNP will_MD be_VB joining_VBP a_DT team_NN already_RB with_PP search_JJ experience_NN>
- Chunk
  - <Jane> <will be joining> <a team> <already with search experience>

- Extract possible links

  Jane => will be joining

  will be joining => a team

  a team => already with search experience

  will be joining => already with search experience

  Jane => already with search experience.

Matrices of Meaning

	tank	team	tent	test	toes	track
jog	0	0	0	0	0	1
jogged	0	0	0	0	0	7
will jog	0	0	0	0	0	5
will be jogging	0	0	0	0	0	4
join	0	3	0	0	0	0
joined	0	8	0	0	0	0
is joining	0	22	0	0	0	0
will be joining	0	10	0	0	0	0
juggle	0	0	0	0	0	0
juggled	0	3	0	0	0	0
will juggle	0	0	0	0	0	0
will be juggling	0	0	0	0	0	0

All about the small

(Tradeoffs between size and performance)

- Netflix dataset ~100 Million triplets
- Our training dataset ~500 Million sentences
- PoS Tagged/Chunked/Pruned
  - Obvious pruning rules makes the problem more tractable
- 200k phrases
  - 200k * 200k = 40 Billion Cells
- And it still wouldn't cover all the cases!
- How can you compress & generalize?

Movie Recommendations

- Collaborative filtering
- Given a matrix of Users and Recommendations
- Predict empty cells
  - Nearest-Neighbor
    - Works poorly for sparse matrix
  - Matrix Factorization
    - Much better on sparse matrix
    - Works via latent feature extraction

	M1	M2	M3	M4
U1	5	3		1
U2	4			1
U3	1	1		5
U4	1			4
U5		1	5	4

Matrix Factorization Collaborative Filtering Basics

- Set *U* of Users
- Set *M* of Movies
- Matrix **R** of size $|U| \times |M|$
- *K* latent features
- Find 2 matrices:
  - **P**($|U| \times K$)
  - **Q**($|M| \times K$)
  - Such that: $\mathbf{R} \approx \mathbf{P} \times \mathbf{Q}^{\mathbf{T}} = \hat{\mathbf{R}}$

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{k} p_{ik} q_{kj}$$

Learning the Factors

$5 = U_{1\text{-}1} * M_{1\text{-}1} + U_{1\text{-}2} * M_{1\text{-}2} + U_{1\text{-}3} * M_{1\text{-}3} \ldots$

$$\min_{q^* * p^*} \sum_{(uj) \in k} (r_{ui} - x_{ui})^2$$

Least Squares Problem

Reduces from 200,000 * 200,000
To 200,000*k + 200,000*k
(k ended up being 200)
40 Billion to 80 Million

	M1	M2	M3	M4
U1	5	3		1
U2	4	?		1
U3	1	1		5
U4	1			4
U5		1	5	4

Rating Prediction

$$\hat{r}_{ij} = p_i^T q_j$$

User Pref. Vector

Movie Pref. Vector

But wait, there's fewer!

Hashing

- Pseudo-random hashing of input phrases
- 50% compression of phrases only cost us 3% of F1 (!!)
- 200,000 phrases now maps to a ~100,000 hash table entries

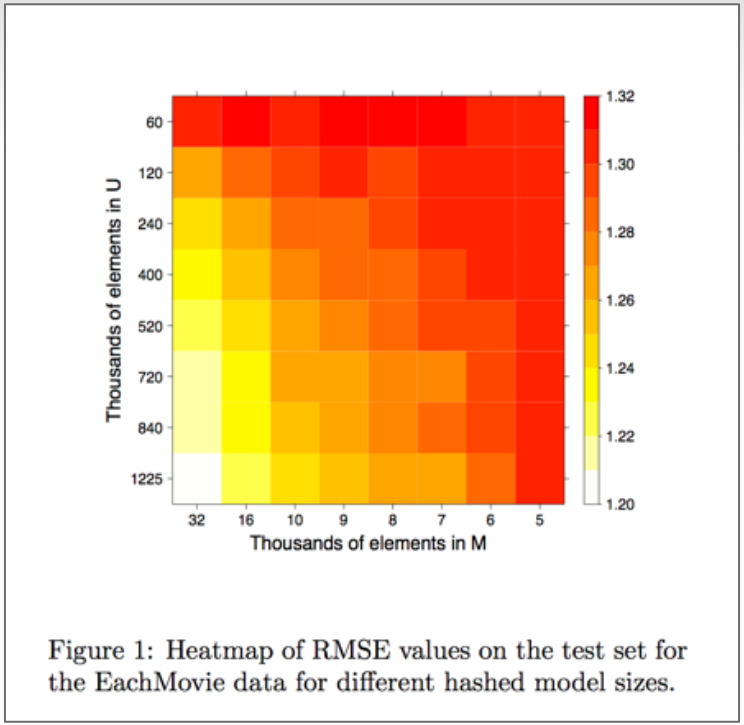
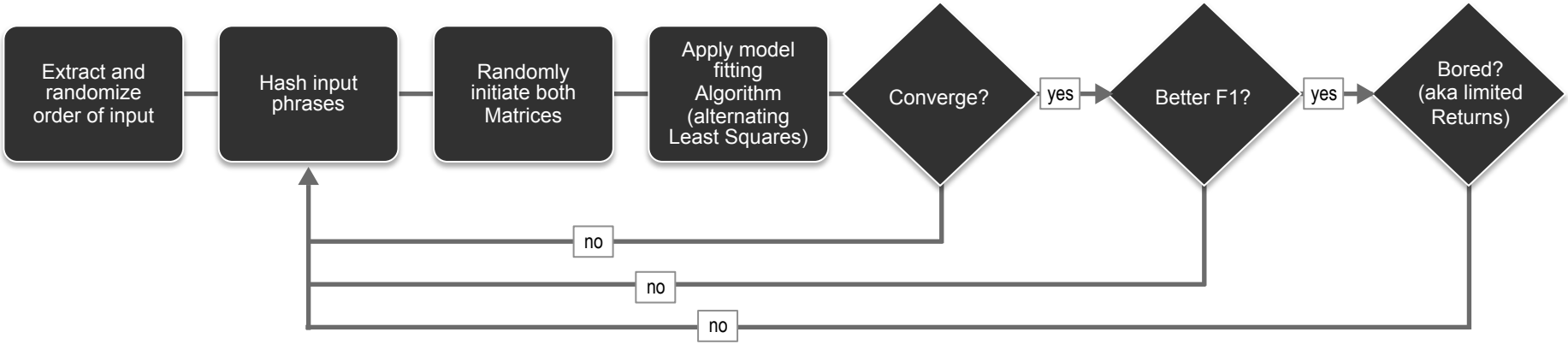


Figure 1: Heatmap of RMSE values on the test set for the EachMovie data for different hashed model sizes.

http://jmlr.csail.mit.edu/proceedings/papers/v9/karatzoglou10a/karatzoglou10a.pdf

Training!



>300 runs until we stopped getting better

Details about Training

- Why Alternating Least Squares vs. Stochastic Gradient Descent?
- Loss Factor: Huber's Robust Loss
- Regularization to avoid overfitting

Huber's robust loss: This loss function (Huber, 1981) ensures robustness for large deviations in the estimate while keeping the squared-error loss for small deviations. It is given by

$$l(f,y) = \begin{cases} \frac{1}{2\sigma}(f-y)^2 & \text{if } |f-y| \leq \sigma \\ |f-y| - \frac{1}{2}\sigma & \text{otherwise} \end{cases}$$

$$\partial_f l(f,y) = \begin{cases} \frac{1}{\sigma}[f-y] & \text{if } |f-y| \leq \sigma \\ \text{sgn}[f-y] & \text{otherwise} \end{cases}$$

Now look at how easy it is

- <Do you want me to get anything else while I go to the store for milk?>
- pos tag and chunk it.
  - <Do> <you> <want> <me> <to get> <anything else> <while> <I> <go> <to the store> <for milk>

Find the possible links.

do want
you want
want me
you to get
want to get
me to get
to get anything else
want while
to get while
while go
I go
go to the store
I to the store
get to the store
want to the store
to the store for milk
go for milk
want for milk

Results

- Reduced “polar opposite” errors by 50%
- Improved worst-case Entity Sentiment F1 scores by 10%
- Improved overall F1 scores
- Great handling of sentences like:
  - Bob’s Haus of Donuts stopped making bad donuts.
  - Rachel stopped playing on time
  - CNN reports that Indiana is facing an unpleasant backlash.


LEXALYTICS
Read Between The Lines