indeed

Spark NLP in Action:

How Indeed Applies NLP to Standardize Resume Content at Scale

Alexis Yelton & Alex Thomas

We help people get jobs.

What is Indeed?

- World’s largest job search engine
- Jobs from many other job sites
- Jobs directly from businesses
- Resumes directly from jobseekers

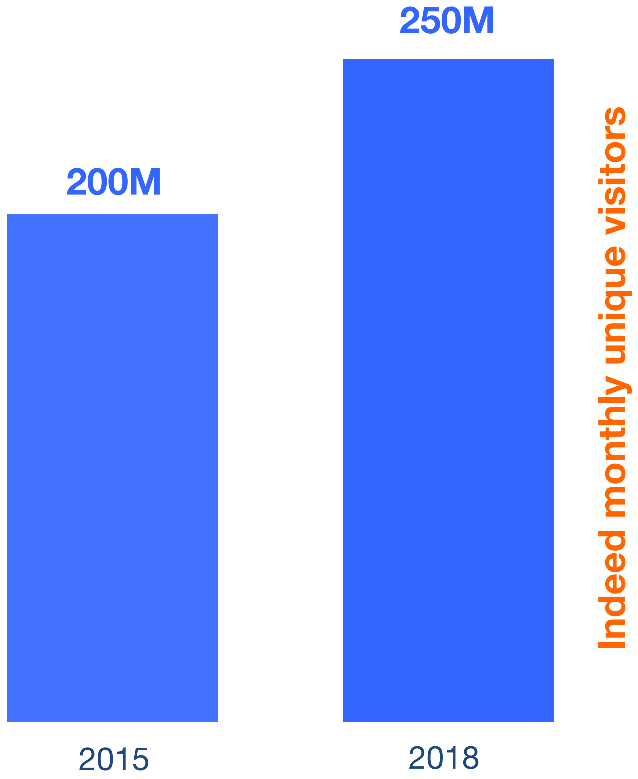
We have Big Data

20 million job descriptions on the site

1 million employers

150 million resumes

100 million reviews



Tools

- [Hive on Spark]
    - Our distributed query engine
    - Data is organized by the products that produce them
- [Apache Spark]
    - DataFrame API for defining ETL
    - Spark MLLib for defining pipelines and ML workflows

Tools

- Spark NLP
  - Open source NLP library with an Apache License. It is implemented using the Spark MLLib framework
  - We are using it for basic text processing, but it also contains implementations for many NLP tasks including PoS tagging, word embeddings (word2vec, GloVe, BERT)

How do we match jobseekers and jobs?

Vocabularies differ

- across industries
- across regions
- between jobseekers and employers

How do we match a job that requires

Massage Therapy License

with resume that has

Licensed Massage Therapist

How are these values distributed?

What is normalization?

How do we understand content?

Semi-structured text data at Indeed is used in models with TF-IDF, deep learning.

Resumes and job descriptions have structure for a reason. How can we interpret the contents of each field?

We need normalization. Otherwise we are stuck with millions of unique values (e.g. bio, biology, boilogy, etc.)

What is normalization?

Classifying terms as a standard term (finding equivalence classes)

Allows for:

- Querying of equivalent terms with search engine
- Creating a small set of features or classes from a corpus to use in modeling
- Deduplication of data and grouping of appropriate entities (e.g. if we want to know how many jobs we have posted for 7-11 we need to look at 7-eleven, 711, etc.)

Examples of skills normalization

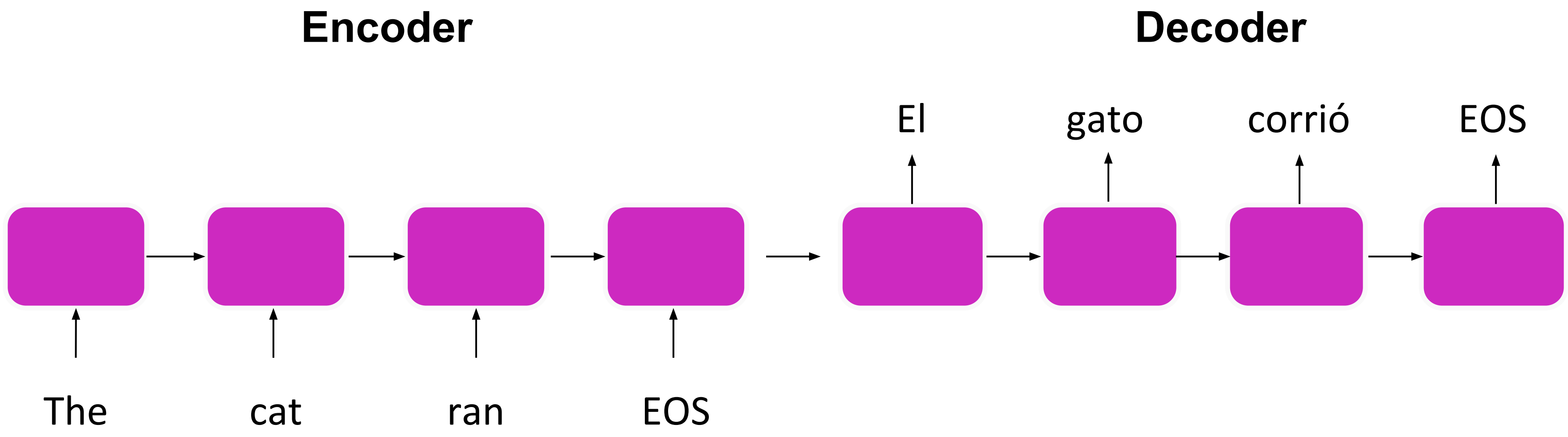
graphic/design editing	**graphic design**
graphic-design tool	**graphic design**
graphic and design	**graphic design**
gimp 2.10 graphic design	**graphic design**
basic graphic designer	**graphic design**
graphic design and drafting	**graphic design**
graphic design-	**graphic design**
graphics designer	**graphic design**

microsoft office sutie	**microsoft office**
microsoft office 10-	**microsoft office**
microsoft office ppt	**microsoft office**
microsoft office proficient	**microsoft office**
microsoft office 2010/2011/2016	**microsoft office**
microsoft --- office	**microsoft office**
microsoft office 2000.	**microsoft office**

Normalization: Methods

- **Rule Based Normalization**
  - Regex
  - Simple rules for capitalization etc.
- **Learned Normalization**
  - Machine translation (Character level statistical machine translation or seq2seq model)
  - Distance metrics (Phrase embeddings, string distance)

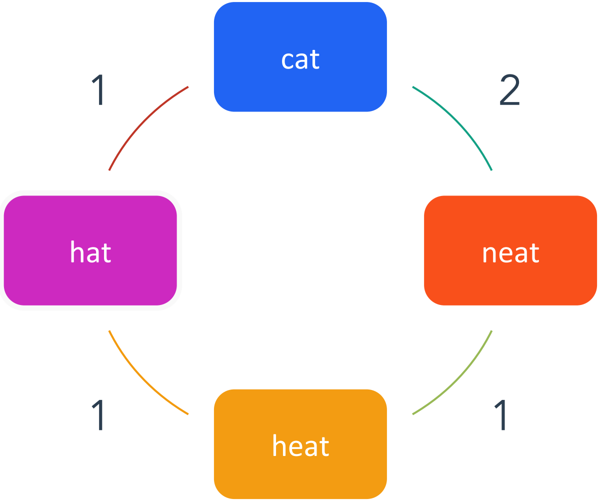
Normalization: Machine translation



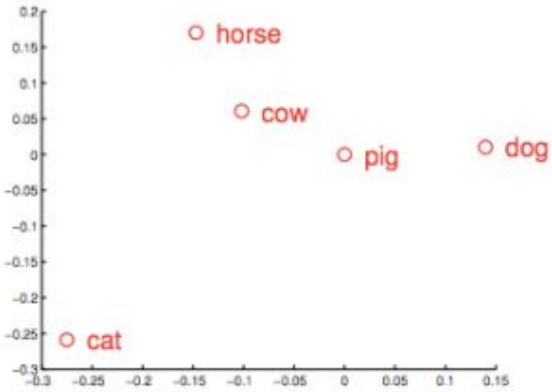
- This can be accomplished with a recurrent neural network
- Probabilities of a normalized value are calculated based on inputs of a corpus ***and a translation of that corpus***

Normalization: Distance metrics

String distance metrics



Vector of features



https://skymind.ai/wiki/word2vec

- Distances can be calculated based on strings themselves or features of the strings (embeddings)

Rule based methods

Advantages

- Human oversight at all steps of the process can prevent errors
- Can capture known situations and catch known exceptions (based on subject area knowledge)

Disadvantages

- Humans are expensive and slow
- Adding new languages requires new rules
- As content changes over time rules need to be updated manually

Model based methods

Advantages

- Fast and cheap
- Can capture unknown situations and catch unknown errors
- Adapts to new languages more easily
- Adapts to content changes more easily

Disadvantages

- Less human oversight can lead to errors (but can add human oversight)
- Less control over the method can lead to errors

StringNormalizer library:
Learned approach

Push of a button

Normalization method

Finds edit distances between strings and groups similar strings (based on multiple features)

Defines a distance vector between strings.

Within each group classifies the unnormalized string as the string with the minimal distance and highest frequency in corpus

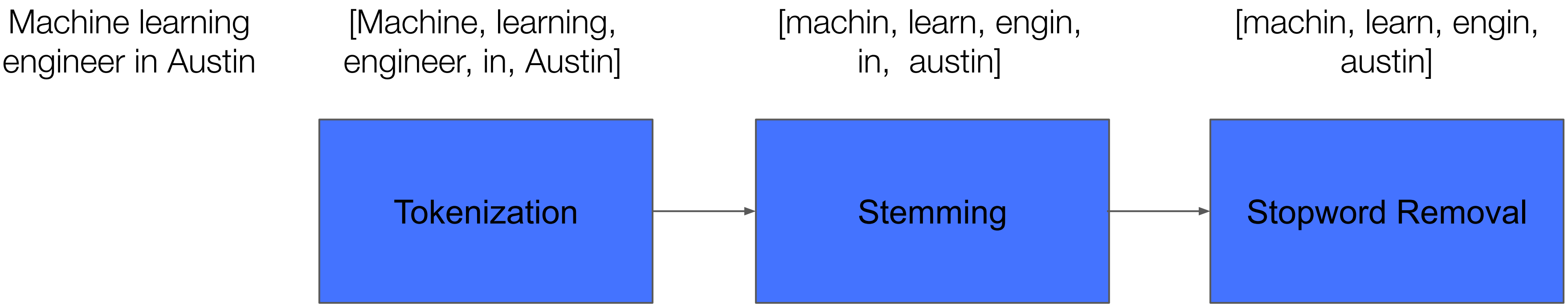
Strengths

- Normalized values to a small percentage of original size
- Reusable design for other types of text
- Can use multiple data sets
- Can find normalized values in whole sentences or lists

Weaknesses

- Similar words and similar characters can be misclassified together (e.g. “applied health” and “allied health”)
- Cannot flag strings that are not of the right type (e.g. “credit hours completed” for major in college)

Preprocessing



Preprocessing: Rules

Acronyms or abbreviations

RN → Registered Nurse

Numbers

7-eleven → 7-11 → seven - one one

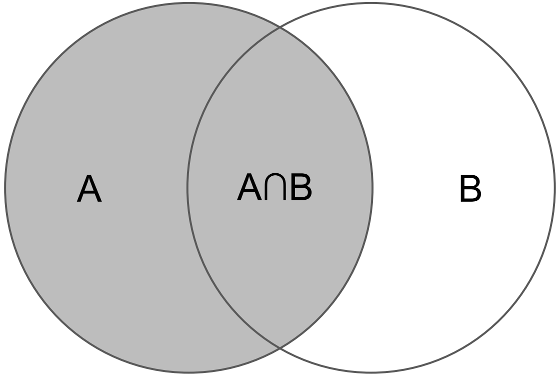
Document frequency filter

Only consider high frequency strings as the normalized value

Does not throw out low count data, but will not use them as normalized values

Jaccard Distance

Based on words



A∩B/A∪B

Business administration and accounting

Business administration and marketing

3/5

MinHash algorithm and Jaccard Distance

Based off of permutations of matrix of fields vs. words

permutation of vocab	business administration and accounting	business administration and marketing
accounting	1	0
business	1	1
and	1	1
administration	1	1
marketing	0	1
minhash value	1	2

Approximation of Jaccard distance with locality sensitive hashing vastly reduces the number of pairwise comparisons needed

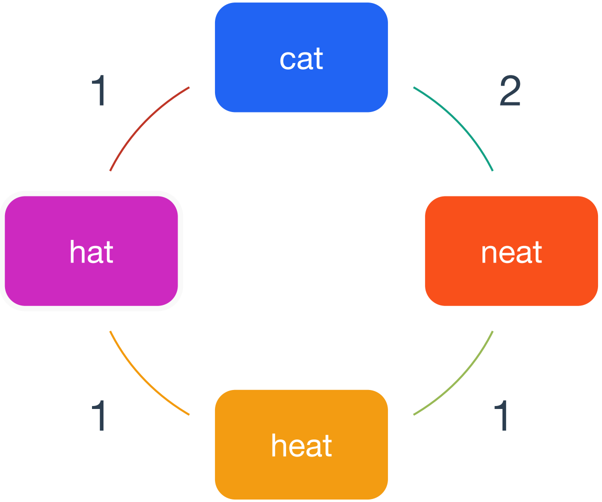
Filters groups of strings to only consider string with J <= X

Levenshtein distance

Number of insertions, deletions, plus substitutions of characters needed to change one string to another

L/maxStringLength gives a normalized Levenshtein distance

Filters data to only consider strings with L <= Y



Majority vote or Euclidean distance classification

Original method:

- Normalized value = value with the highest number of counts

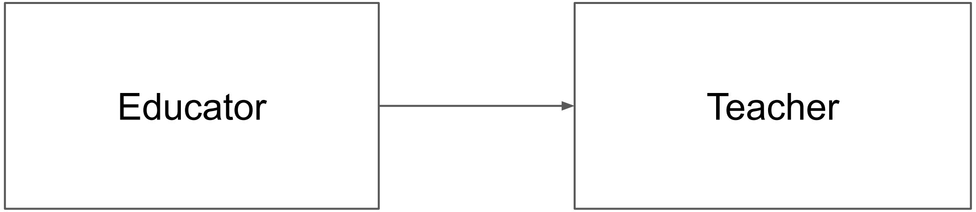
Refined method:

- C = Weight/Counts
- Vector of J, L, C
- Normalized value = minimum L2 norm among vectors → Euclidean distance

Allows for exact matches for rarer values to be chosen over imperfect matches, but common values are generally chosen

Post-processing: Rules

Mapping with existing regex or known synonym sets



Pipeline

Preprocessing → Document frequency → MinHash → Ldistance → Normalization

Skills	Document frequency >= 200	MinHash (Jdistance <= 0.5)	Ldistance <= 0.3	Euclidean distance
I love Lucy	1			
customer service experience	200	customer service experience	customer service experience	customer service experience
customer service rep	100	customer service rep	customer service rep	customer service rep
auto mechanic	1200			
online customer service	200	online customer service		

Thresholding/Tuning

Counts	Method	Accuracy	% normalized total fields	% normalized unique values
100	Euclidean	89%	89%	0.42%
200	Euclidean	91%	87%	0.24%
500	Euclidean	91%	83%	0.11%

Num Counts affects granularity, not accuracy for Euclidean method

Summary

- Normalization provides an automated way to standardize content
- For big data, Spark and SparkNLP provide an easy way to process content for normalization
- String distances and frequency can automatically generate normalized strings within a corpus
- Some human intervention is ideal (model tuning and pipelined rules)

Thank you