Modern approaches to Entity Resolution

• Ph.D. Course, Roma Tre University, Spring 2020

• Donatella Firmani
• donatella.Firmani@uniroma3.it

http://inf.uniroma3.it/~firmani/coursephd
Modern Approaches for Recognition of Duplicates

Lecture 2
Outline

• Basic notions

• A design space for duplicate recognition systems with high F-measure
Basic Notions
String Similarity

- Syntactic similarity of different representations of the same entity
- Canon ~ Canon Inc.
- Canon ~ Canoon
Popular Metrics

Edit Based

- Soundex
- Levenshtein / edit distance
- Jaro / Jaro-Winkler

Term based

- Jaccard Coefficient
- Tf-idf
- Cosine similarity

N. Koudas, S. Sarawagi, D. Srivastava: Record linkage: similarity measures and algorithms. SIGMOD 2006: 802-803
Edit Based: Soundex

- A phonetic algorithm that indexes names by their sounds when pronounced in English.
- Consists of the first letter of the name followed by three numbers. Numbers encode similar sounding consonants.
  - Remove all W, H
  - B, F, P, V encoded as 1, C,G,J,K,Q,S,X,Z as 2
  - D,T as 3, L as 4, M,N as 5, R as 6, Remove vowels
  - Concatenate first letter of string with first 3 numerals

- Example: great and grate become 6EA3 and 6A3E and then G63
Edit Based: Edit Distance

- Given two strings, $s,t$:
  - Character Operations: I (insert), D (delete), R (Replace).
  - $\text{edit}(s,t)$: Length of the shortest sequence of operations to transform $s$ to $t$.

- Example: $\text{edit}(\text{Error},\text{Error}) = 1$, $\text{edit}(\text{great},\text{grate}) = 2$

- Computation: quadratic (on string length) in the worst case.
Edit Based: Jaro

- Given two strings, s,t, edit(s,t):
  - s’: characters in s that also appear in t, not farther than H = min(|s|,|t|)/2
  - t’: characters in t that also appear in s, not farther than H = min(|s|,|t|)/2
  - Transpositions T: Positions in s’ and t’ with different characters (divided by two)
  - edit(s,t): \( \frac{1}{3} \left( \frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'|-T}{|s'|} \right) \)

- Example:
  - Martha vs Marhta
    - H=3, s’=Martha, t’=Marhta, T = 1
    - Jaro(Martha,Marhta) = 0.9722
  - Jonathan vs Janathon
    - H=4, s’=Jnathn,t’=Jnathn, T = 0
    - Jaro(Jonathan,Janathon) = 0.5
Term Based: Jaccard

- Terms (Unordered)
  - Words: ‘AT&T Corporation’ -> {‘AT&T’, ‘Corporation’}
  - Q-grams (sequence of q-characters): 3-grams {‘AT&’,‘T&T’,‘&T’, ‘T C’, ‘Co’,‘orp’,‘rpo’,‘por’,‘ora’,‘rat’,‘ati’,‘tio’,‘ion’}

- Jaccard Coefficient
  - Given two sets of terms S, T
    - Jaccard(S,T) = |S∩T|/|S∪T|
  - If scores (weights) available for each term we can compute Jaccard() only for terms with weight above a specific threshold.
Term scoring

- Term frequency (tf) inverse document frequency (idf).
- Widely used in traditional IR approaches.

- The tf/idf value of a ‘term’ in a document: log (tf+1) * log idf
  - tf : # of times ‘term’ appears in a document d
  - idf : number of documents / number of documents containing ‘term’

- Use the string as a document and the set of string values as the document collection.
- Intuitively: rare ‘terms’ are more important
Term Based: Cosine

- Each set of terms S is transformed to a sparse vector $S$ of high dimensionality.
  - via TF-IDF score of its terms
  - via TF score of its terms

- Given two sets of terms S, T
  - $\text{Cosine}(S,T) = S \cdot T / \| S \| \| T \|$
  - Where $S \cdot T = \sum_{z \in S \cap T} S[z] T[z]$ and $\| S \|$ denotes the Euclidean norm
  - $\text{Cosine}(S,T)$ denotes the cosine of the angle between S and T
    - $-1$ means opposite, $1$ means the same, $0$ means orthogonality or decorrelation
Word Embedding

- Semantic similarity of different representations of the same entity
- Canon Inc. ~ Kyanon kabushiki gaisha
- CMOS ~ Complementary Metal-Oxide Semiconductor
Intuition

- You can get a lot of value by representing a word (or a term) by means of its neighbors.
  - Possibly from a separate corpus independent from the ER task

- “You shall know a word by the company it keeps” (J. R. Firth 1957: 11)

```plaintext
政府债务问题正在转化为银行危机，正如在欧洲发生的那样，需要统一的银行监管来取代混乱。

这些词语将代表 banking
```
In a nutshell

A word embedding model maps terms or phrases from a vocabulary to vectors of real numbers. Methods include neural networks and dimensionality reduction techniques. Details outside the scopus of this course. Typically pre-trained and used in combination with specific application.


R. Lebret et al. Word Embeddings through Hellinger PCA. EACL. 2014

Example: Word2Vec embedding model

Define a model that predicts between a word $w_i$ and context words

- Skip-gram: predicts surrounding words given the current word $p(\text{context} | w_i)$
- Bag-of-Words: predict the current word based on the context $p(w_i | \text{context})$

Look at many positions $i$ in a big language corpus

- possibly independent of the ER task

Loss function, e.g., $J=1-p(w_{i+j} | w_i)$

Keep adjusting the vector representations of words to minimize this loss
Word2Vec Skip-gram

• Given a sentence and a center position i
  • c=w_i is the current word
  • t =w_{t-1}, w_{t+1}, w_{t-2} ... is a word in the context that I want to predict via p (t | c)

https://simons.berkeley.edu/sites/default/files/docs/6449/christophermanning.pdf
https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b
Word2Vec Skip-gram

• Input vector can be one-hot
• Words occurring in different contexts of the same word are driven to have similar vectors

\[ p(t|c) = \frac{\exp(\text{embedding}(t)^T \text{input\_vector}(c))}{\sum_{w \in \text{vocabulary}} \exp(\text{embedding}(w)^T \text{input\_vector}(c))} \]

https://simons.berkeley.edu/sites/default/files/docs/6449/christophermanning.pdf
https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b
Comments

• Embedding space
  • Cosine captures similarities
  • Linear vector differences capture semantic concepts

• Out-Of-Vocabulary Problem
  • In principle can use terms such as q-grams instead of words to solve this problem
  • Other models such as FastText and Glove do not have this problem

Pennington, J., Socher, R., & Manning, C. D. "Glove: Global vectors for word representation." EMNLP 2014
Match synonyms like FullHd with 1080p

Option 1

- Keep a simple (sparse) vector representation e.g. one-hot
  - \( \text{FullHd} = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0] \)
  - \( \text{1080p} = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0] \)
- Cosine(\( \text{FullHd} \), \( \text{1080p} \)) = 0
- Learn metric semantic_similarity(\( \text{FullHd} \), \( \text{1080p} \))

Option 2 (win)

- Learn a (dense) word representation that encodes similarity
  - \( \text{1080p} = [0.1 \ 0.4 \ 0.5 \ 0.2 \ 0.1 \ 0.3] \)
  - \( \text{FullHd} = [0.2 \ 0.8 \ 1.1 \ 0.4 \ 0.3 \ 0.6] \)
- Use simple metric, e.g., Cosine(\( \text{FullHd} \), \( \text{1080p} \)) = 0.997
- Learn less parameters (per word, not pair)
Duplicate Recognition
Main Concepts

01 Use **word-embedding** to represent tokens in the dataset

02 Reuse well known techniques for nlp processing to **summarize & compare attribute values**

03 Use a classification layer to learn matching and non-matching relations

04 Use **string similarity** to generate non-trivial negative examples for training
01 Token representation

• Train a word embedding model e.g. word2vec, glove or fasttext
  • Take any informative corpus, such as Wikipedia
  • Or download a pre-trained model by somebody else

• Represent a tuple as follows
  • For all attributes take the sequences of words (or - more in general – terms) and convert them to sequences of d-dimensional word embeddings (es d=2)

<table>
<thead>
<tr>
<th>Brand</th>
<th>Model</th>
<th>Megapixel</th>
<th>Sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canon</td>
<td>4000D</td>
<td>18.0 Mp</td>
<td>NULL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Brand</th>
<th>Model</th>
<th>Megapixel</th>
<th>Sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canon</td>
<td>4000D</td>
<td>17.9</td>
<td>CMOS</td>
</tr>
</tbody>
</table>

02 Attribute Summarization & Comparison

• Input: embeddings for corresponding attribute value pairs
• Output: encodes this input to a representation that captures the attribute value similarities of e1 and e2
Comments

1. We are assuming that schema is perfectly aligned
   • If it is not the case (or if partially is) you can merge into a single attribute
   • In practice, this is like performing a “trivial” schema-alignment

<table>
<thead>
<tr>
<th>Brand</th>
<th>Model</th>
<th>Megapixel</th>
<th>Sensor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canon</td>
<td>4000D</td>
<td>17.9</td>
<td>CMOS</td>
<td>Canon 4000D 17.9 CMOS</td>
</tr>
</tbody>
</table>

   • State-of-the-art methods typically do not deal natively with schema alignment and record linkage together

2. Word embedding layer cannot be fine tuned (input layer)
02 Attribute Summarization & Comparison

• Two steps
  1. Each cell needs to be summarized as a fixed-length single vector (e.s. 18.0 Mp)
  2. Represent similarity of pairs of summarized cells

• Step 1
  • Core component
  • Choose between different layers (varying representational power)
    • Fully connected (naive)
    • Recurrent Neural Network
    • Attention
    • ...

• Step 2
  • Choose between different vector similarity metrics
    • Cosine
    • Euclidean
    • ...

Recurrent Neural Networks

Input: a sequence of vectors \( x_1, x_2 \ldots x_m \)

Output: a sequence of vectors \( y_1, y_2 \ldots y_m \)

RNN are great at processing sequences (e.g. speech recognition)

RNN processes one vector at a time from left to right (bidirectional RNNs are also available)

\( y_i \) is a function of \( x_1 \ldots x_i \)

\( y_m \) can thus provide a "summary" of the entire input sequence

More in detail:

Hidden state vector updated at every step

At step i: take the i-th vector and the hidden state output of the previous time step \( i-1 \)

The hidden state vector of the i-th time step contains information from inputs \( x_1 \ldots x_i \)

\( y_i \) is set to the i-th hidden state output
Attention

• Traditional RNNs
  • compress the input sequence to a fixed-length vector representation
  • consider all parts of the input sequence to be equally important
  • long and noisy sequences can be hard to compress meaningfully

• Attention mechanism can be combined to RNN to overcome this issue
  • For instance, by giving more importance to the model name in a long descriptive attribute

Description

Canon 4000D
17.9 CMOS
03 Classification

- multi-layer perceptron
04 Training

• Ground truth consisting of positive and negative pairs
• In DeepER the negative training samples (pair of non-matching records) are built in the following way:
  ● Let S1 and S2 the two sources of records
  ● Take random pairs from S1,S2 and evaluate their similarity with common string similarity functions, such as Levenshtein, Jaro-Winkler, Jaccard etc..
  ● If the similarity of a pair is under a threshold T the pair is considered a negative sample

This technique allows to increase F-measure significantly

# F-measure

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RNN</th>
<th>Attention</th>
<th>Magellan</th>
</tr>
</thead>
<tbody>
<tr>
<td>iTunes-Amazon2</td>
<td>79.4</td>
<td>63.6</td>
<td>46.8</td>
</tr>
<tr>
<td>DBLP-ACM2</td>
<td>97.5</td>
<td>97.4</td>
<td>91.9</td>
</tr>
<tr>
<td>DBLP-Scholar2</td>
<td>93.0</td>
<td>92.7</td>
<td>82.5</td>
</tr>
<tr>
<td>Walmart-Amazon2</td>
<td>39.6</td>
<td>53.8</td>
<td>37.4</td>
</tr>
<tr>
<td>Home3</td>
<td>86.4</td>
<td>88.0</td>
<td>68.6</td>
</tr>
<tr>
<td>Tools3</td>
<td>92.8</td>
<td>92.6</td>
<td>76.1</td>
</tr>
</tbody>
</table>

Other approaches

- **Bert + Classifier**
  - direct application of Transformer models (e.g. the BERT family)
    - highly-contextualized (can capture that the word Sharp has different meanings in “Sharp resolution” versus “Sharp TV”)
    - can solve high-level NLP tasks such as next sentence prediction
  - two records serialized as one sequence and fed to the pre-trained model
    - simplification: akin to computing a word embedding of the entire pair
    - learn a binary classifier

- **Task-specific Embeddings**
  - word embeddings models over corpuses like Wikipedia may not work for dataset with custom vocabulary (e.g., 1080p)
  - solution: derive sentences that effectively “describe” the similarity across elements (tokens, attributes, rows) from two sources.
    - pre-train word embedding model on the resulting corpus


R. Cappuzzo, P. Papotti, S. Thirumuruganathan: Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks. SIGMOD 2020: 1335-1349
Conclusions

- Discussed basic notions to compute similarity
  - String similarity
  - Word embedding

- Presented a design space to implement high f-measure duplicate recognition