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).
  - edit(s,t): Lenght of the shortest sequence of operations to transform s to t.

- Example: edit(Error,Eror) = 1, edit(great,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): 1/3 (|s'|/|s| + |t'|/|t| + (|s'|-T)/|s'|)

- 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) = $|S \cap T| / |S \cup T|$

  - 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
  - Cosine(S,T) = **S T** / ||**S**|| ||**T**||
  - Where **S T** = $\sum_{z \in S \cap T}$ ***S[z] T[z]*** and ||S|| denotes the Euclidean norm
  - 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)

government debt problems turning into banking crises as has happened in

saying that Europe needs unified banking regulation to replace the hodgepodge

↖ These words will represent *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

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. "A Neural Probabilistic Language Model ". JMLR (2003), 1137–1155.

T. Mikolov, I. Sutskever, K. Chen, et al. Distributed Representations of Words and Phrases and Their Compositionality. NIPS. 2013

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

A. Globerson, G. Chechik, F. Pereira, and N. Tishby. Euclidean Embedding of Co-occurrence Data. JMLR 8 (2007), 2265–2295.

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(context | w_i)
- Bag-of-Words: predict the current word based on the context p(wi | context)

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

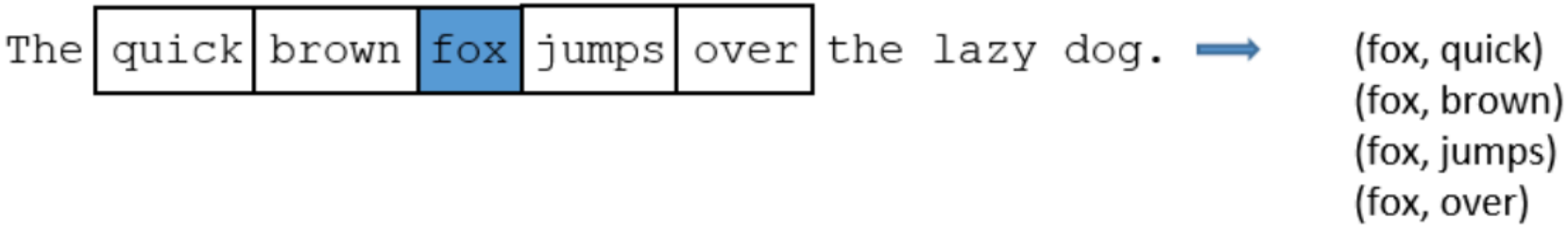
- possibly independent of the ER task

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)



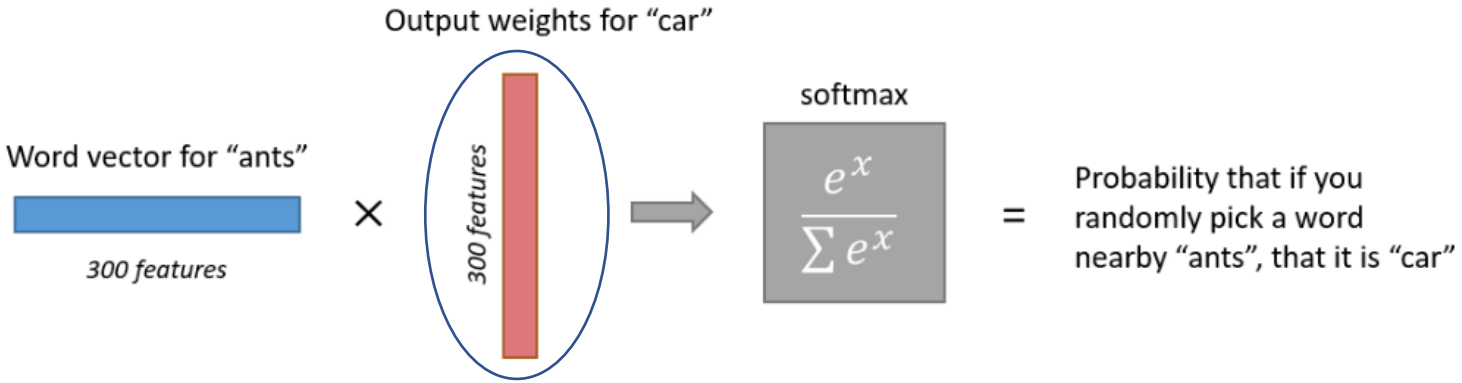
https://simons.berkeley.edu/sites/default/files/docs/6449/christophermanning.pdf

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

$$\mathrm{p(t|c)} = \frac{\exp(embedding(t)^T input\_vector(c))}{\sum_{w \in Vocabulary} \exp(embedding(w)^T input\_vector(c)}$$

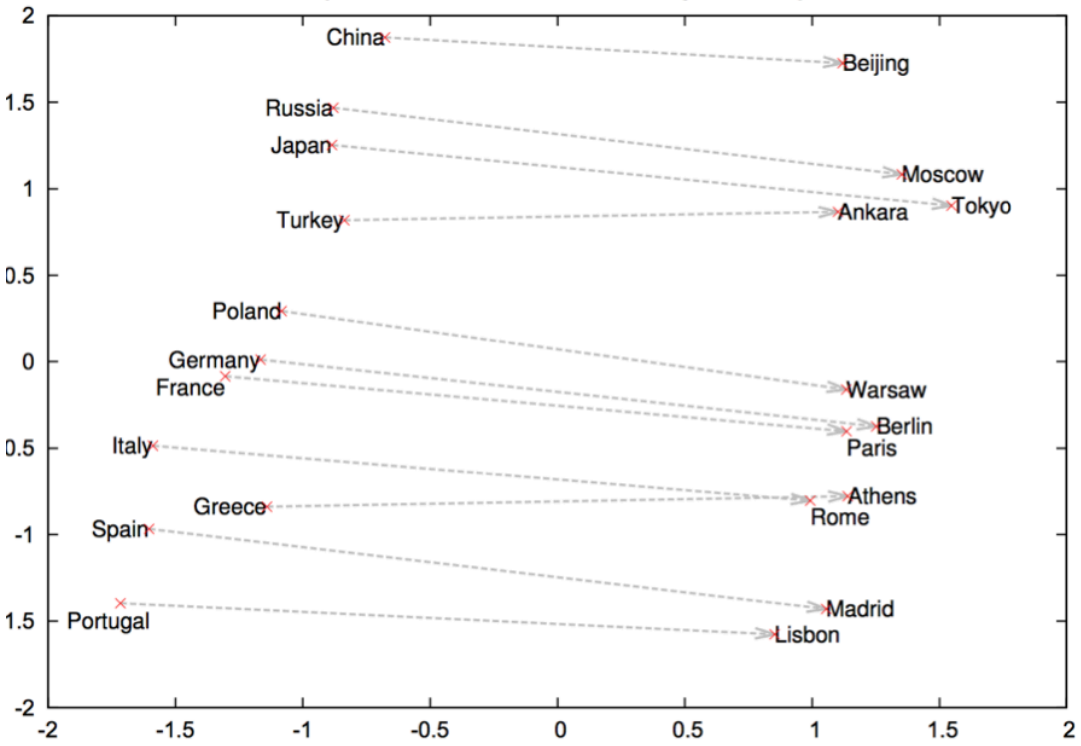


https://simons.berkeley.edu/sites/default/files/docs/6449/christophermanning.pdf

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



Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, Huanbo Luan. Joint Learning of Character and Word Embeddings. JCAI 2015.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T.. "Enriching word vectors with subword information." TACL, *5*, 135-146. 2017.

Pennington, J., Socher, R., & Manning, C. D. "Glove: Global vectors for word representation." EMNLP 2014

Motivation

Match synonyms like FullHd with 1080p

Option 1

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

Option 2 (win)

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

Christopher Manning. "Representations for Language: From Word Embeddings to Sentence Meanings». 2017

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)

S1

Brand	Model	Megapixel	Sensor
Canon	4000D	18.0 Mp	NULL

Brand	Model	Megapixel	Sensor
[0.3 0]	[3 8.7]	[2.2 3.0] [3 7]	[0 0]

S2

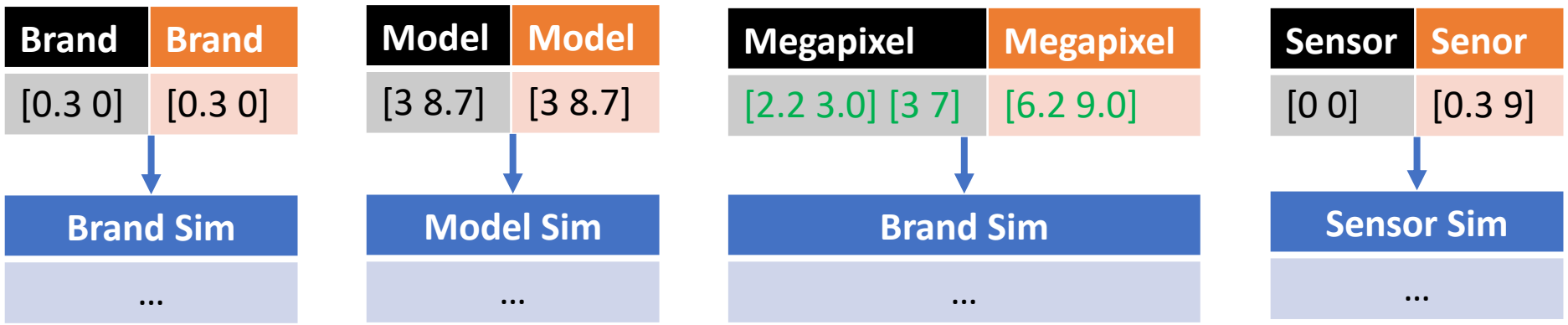
Brand	Model	Megapixel	Sensor
Canon	4000D	17.9	CMOS

Brand	Model	Megapixel	Sensor
[0.3 0]	[3 8.7]	[6.2 9.0]	[0.3 9]

S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, V. Raghavendra: Deep Learning for Entity Matching: A Design Space Exploration. SIGMOD Conference 2018: 19-34

02 Attribute Summarization & Comparison

- Input: embeddings for corresponding attribute value pairs
- Ouptut: 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

Brand	Model	Megapixel	Sensor
Canon	4000D	17.9	CMOS

→

Description
Canon 4000D 17.9 CMOS

   - 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 similariy metrics
    - Cosine
    - Euclidean
    - …

Recurrent Neural Networks

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

Input: a sequence of vectors x_1, x_2 … x_m

Output: a sequence of vectors y_1, y_2 … y_m

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

y_i is a function of x_1 … 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 … x_i .

y_i is set to the i-th hidden state output

Attention

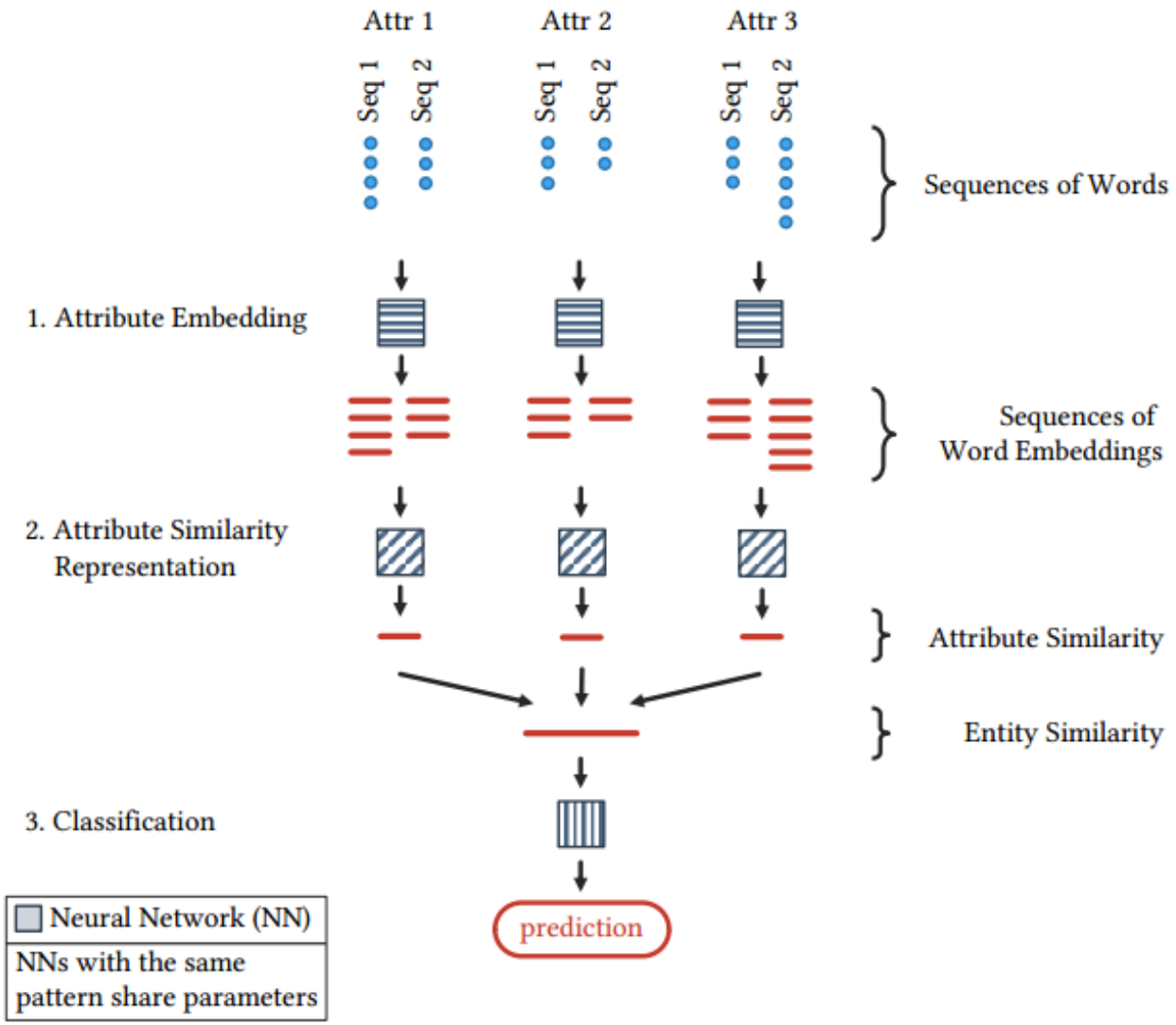
- Traditional RNNs
  - compress the input sequence to a fixed-length vector representation
  - consider all parts of the input sequence to be equally important
  - long and noisy sequences can be hard to compress meaningfully

- 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

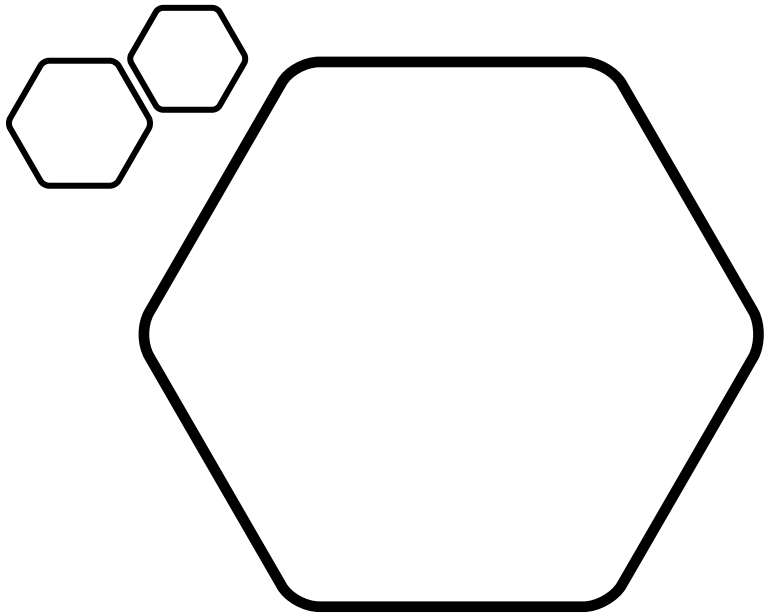
Description
Canon 4000D 17.9 CMOS

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

M. Ebraheem, S. Thirumuruganathan, S. R. Joty, M. Ouzzani, N. Tang: Distributed Representations of Tuples for Entity Resolution. Proc. VLDB Endow. 11(11): 1454-1467 (2018)

F-measure

Dataset	RNN	Attention	Magellan
iTunes-Amazon2	**79.4**	63.6	46.8
DBLP-ACM2	97.5	**97.4**	91.9
DBLP-Scholar2	**93.0**	92.7	82.5
Walmart-Amazon2	39.6	**53.8**	37.4
Home3	86.4	**88.0**	68.6
Tools3	**92.8**	92.6	76.1

S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, V. Raghavendra: Deep Learning for Entity Matching: A Design Space Exploration. SIGMOD Conference 2018: 19-34

Other approaches

- Bert + Classifier
  - direct application of Transformer models (e.g. the BERT family)
    - highly-contextualized (can capture that he 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

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv:1810.04805 (2018).

Y. Li, J. Li, Y. Suhara, A. Doan, W. Tan: Deep Entity Matching with Pre-Trained Language Models. CoRR abs/2004.00584 (2020)

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