UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

IHLT

Semantic Textual Similarity

Albert Rial

Utku Ünal

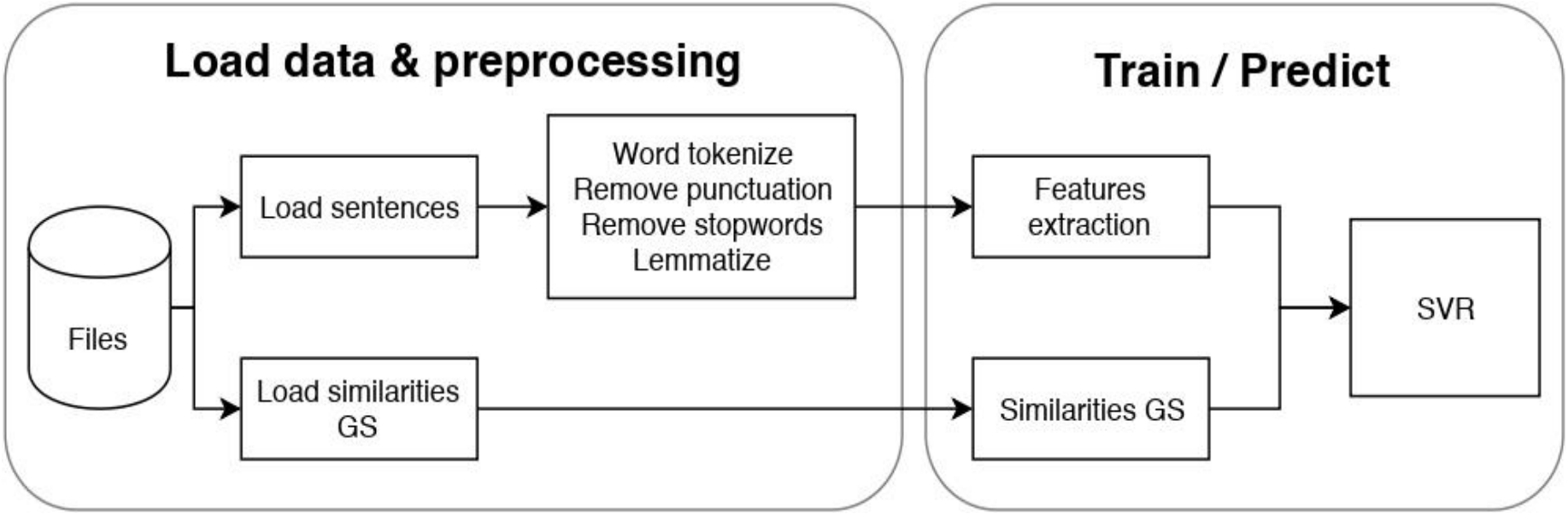
Master in Artificial Intelligence

19/12/2019

Introduction

- Task of Semantic Textual Similarity (STS) included in the SemEval
- Consists in given two pair of sentences, provide a similarity value between them

Our system


Load data & preprocessing
Train / Predict
Files
Load sentences
Word tokenize
Remove punctuation
Remove stopwords
Lemmatize
Load similarities GS
Features extraction
Similarities GS
SVR

Beyond the Scope

- First system
  - Used concepts and techniques learnt in the subject
- Second system, in order to try to overpass the team in first position:
  - Previous techniques + Pre-trained model of sentence embeddings (InferSent)

Preprocessing

- Word tokenization
- Transform to lowercase
- Removing punctuation
  - !"#$%&'()*+, -./:;<=>?@[\]^_`{|}~)
- Removing stopwords
- Getting lemmas
- Computing word relevance:
  - Total number of words / Frequency of each word
  - Words that appear less in corpus have more weight (more meaning)

Process - 1

- Started with very simple features:
  - Jaccard similarity between lemmas and words
  - Cosine similarity between lemmas and words
  - LCH similarity between synsets
- Thought about using CNN, but corpus too small
- Decided to implement a linear regression using Support Vector Regression

Correlation: **61.8%**

Process - 2

- Next features:
  - Count if sentences had the same number of entities of the same kind
  - Length difference
  - Sentiment polarity similarity
  - Lesk synsets similarity

Correlation: **63.2%**

Process - 3

- We saw most important feature was LCH Similarity between Synsets
- Decided to implement also:
  - Path Similarity
- Tuned hyperparameters of SVR:
  - Kernel: Radial basis function kernel
  - Gamma, regularization parameter, epsilon and tolerance

Correlation: **66.4%**

Process - 4

- Analyzed in which sentences we were failing
- New features:
  - Unigram similarity
  - Bigram similarity
  - Trigram similarity
- Used dice similarity in these features

$$S_{dice}(X,Y) = \frac{2 \cdot |X \cap Y|}{|X| + |Y|}$$

Correlation: **68.2%**

Process - 5

- Using the relevance of words, added:
  - Unigram similarity giving more weight to the important unigrams/words

Correlation: **70.6%**

Process - 6

- Improved the synset similarity methods by:
  - Considering all the synsets of each lemma (not only the most frequent)
  - Similarity of (s1, s2) is not the same as similarity of (s2, s1) → computed both
- Also added:
  - Lin Similarity between Synsets
  - Wup Similarity between Synsets

Correlation: **75.7%**

Final process

- Analyzed the importance of each feature
- Removed unimportant features
- Tuned hyperparameters of SVR

Final process

Kept:

- Jaccard similarity
- Synsets similarity (all)
- Length difference
- Unigrams, weighted unigrams, bigrams and trigrams similarity

Discarded:

- Cosine similarity
- Count number of entities
- Lesk synsets similarity
- Sentiment polarity similarity

Correlation: **78.02%**

Second system

- Trying to reach first place
- Removed all our features and added only one feature:
  - Euclidean distance between sentence embeddings given by InferSent
- Trained with 1.000.000 most common words of English and words in our corpus.

Correlation: **75.5%**

Second system

- InferSent feature + some of our features:
  - Jaccard similarity
  - Cosine similarity
  - Synsets similarity (path and wup)
  - Length difference
  - Unigrams, weighted unigrams, bigrams and trigrams similarity

Correlation: **82.46%**

Conclusions

✓ With simple techniques and features learnt in subject → high correlation (8th place and result very close to 5th)

✓ Using pre-trained model of sentence embeddings → overpass the result of the winner of SemEval 2012

✗ Efficiency → 4-inner loop when comparing all synsets of all lemmas

Thank you

Albert Rial

Utku Ünal