BIG DATA FOR AUTOMATIC RELATION EXTRACTION IN NATURAL LANGUAGE PROCESSING

Using Word Embedding and Word2vec

Serre Jérémym
November 2017
Supervised by Dr. Prof. Philippe Cudré-Mauroux & Alisa Smirnova

eXascale Infolab in University of Fribourg
GOALS

• Extract relations from raw corpus pairs of words (« Paris - France ») using Word2Vec.

• Generate new pairs with the same relation type given in input.

• Evaluate and measure the reliability of the retrieved pairs.

• Focus on improving the precision of the retrieved pairs.

• Improve the computation time.
CHALLENGES

➤ Extract relation from unlabeled big data corpus.

➤ Starting from an existing undocumented program (Matúš Pikuliak) which runs on a single machine.

➤ Using words embedding for extracting pair relations.

➤ Work in a distributed environment
OUTLINE

➤ Pre-process a big data corpus from "Wikipedia dump" (General Field).
➤ Use the pre-processed corpus in order to create a Word2Vec Model.
➤ Deploy the relation extraction program from Gensim to Spark
➤ Select the pairs in input of the RE program with our new selection methods.
➤ Extract relations with our algorithm using the Word2Vec model.
➤ Evaluate the relations in an automatic way with a Knowledge Base (KB).
➤ Measure scores of the results (precision/nDCG) of these relations and compare them.
➤ Compare the execution time.
INTRODUCTION
INTRODUCTION
INTRODUCTION
MAIN TECHNOLOGIES USED

➤ **Hadoop** for the Distributed File system HDFS.

➤ **Yarn** for the resource management (included in hadoop).

➤ **Spark** for the execution of our algorithms in a distributed environment (using hdfs).

➤ **Gensim** framework for the preprocessing tools.

➤ **Word2Vec** with **Spark** (MLLIB) and the Gensim implementation.

➤ **Wikidata** is a Free Knowledge Database (KB), more precisely a document-oriented database for Semantic Web.
This algorithm produces word embeddings.

Words from corpus are mapped to vectors in multi-dimensional space of real numbers. Each word is positioned in function of its context in the corpus.

CBOB and Skip-gram architecture models.
PAIR STRUCTURE

- A pair is composed of 3 embedding instances.
- An embedding instance is composed of one word and its vector representation.

Embedding 1
Word: "Paris"
Vector: Word Vector

Embedding 2
Word: "France"
Vector: Word Vector

Embedding 3
Word: Concatenation of words from embedding 1 and 2 = "Paris - France"
Vector: Word Vector from embedding 2 - Word Vector from embedding 1
APPROACH

➤ Pre-processing
➤ Fitting the Word2Vec model
➤ Input Pair Selection Methods
➤ Relation extraction
➤ Evaluation
➤ Results comparison
PRE-PROCESSING

➤ Select a global corpus from wikidata.
➤ Remove the XML Wikidata template.
➤ Transform upper case letters to lower case letters.
➤ Remove accent on letters.
➤ Remove non-ASCII characters.
➤ N-Gram (bi-gram, tri-gram and quadri gram).
➤ Stopword Lists.
FIT WORD2VEC MODEL

➤ Vector Size (Number of neurons)
➤ Min Count
➤ Window Size (Context)
INPUT PAIR SELECTION

➤ Some input pairs are close in the multi-dimensional space.
➤ During the generation of neighbours we will obtain almost the same result for closest pairs.
➤ One of the objectives is to obtain a high precision and nDCG score with fewer input pairs as possible.

➤ 4 Methods:
  ➤ Word Count Selection
  ➤ Cosine and Euclidean Input Pair Selection
  ➤ K-Means Selection
INPUT PAIR SELECTION – PART 1

1. **Set the number of groups K.**
2. **Set the number and the size of groups.**
3. **Apply Word Count algorithm on the original corpus.**

**Pairs Input:**
- P1
- P2
- P3
- P4
- P5
- P6
- P7
- P8
- P9

**Step 1:**
- **Create the K-Means Model**
- **Generation of all possible group combinations**
- **Transform all pairs into a tuple of 2 words ((word1, word2), Pair)**

**Input/output of step 1:**
- K-Means Model
- List of lists
- List of tuples

**Step 2:**
- **Classify all the input Pairs with the K-Means Model**
- **The Sum of the similarity score, between pairs of the same group**
- **Add the occurrence of each word in the pair**

**Example of Dictionary of List in our case:**
- Group 1: P1 P7
- Group 2: P3 P4 P6 P9

**List of Tuples composed of a subList and its score. All subList has the same size defined in the first step (Ex: with a size of 3).**
- Group 1: P2 P5 P6 (Similarity Score)

**List of Tuples composed of one pair and its occurrence score:**
- P1 Occurrence Score
- P2 Occurrence Score

... until the number of input pairs
INPUT PAIR SELECTION – PART 2

For each group in the dictionary, compute the SSE score of each pair (Sum of Squared Errors between each element of a group).

Sort all lists by their score in the tuple

Dictionary of list of tuples (Pair, SSE Score)

Rank each tuple by the SSE score in order to obtain the best pairs at the top of the list

Select groups with the highest similarity score and remove the groups with duplicates.

Return the number of pairs wanted by the user

Dictionary of ranked lists

List of lists

All group elements are ranked by their similarity with the other group elements.

List of lists

Ranking list of pairs

K-Means Method

Pair Cosine / Euclidean Selection Method

WordCount Selection Method

Return the number of pairs wanted by the user, selected by the best represent of each group
PAIRS EXTRACTION

- Pairs used as Input for the algorithm
- Generate neighbours of each pair word and perform a Cartesian product between them.
- Each pair from the output list is compared to all the input pairs using a Euclidian similarity.
EVALUATION

- Use a knowledge Base with Wikidata for binary evaluation
- Manual validation for more complex relation (e.g., Genre):
  - 2 : True relation (Barman - Waitress)
  - 1 : Half True relation (Bartender - Waitress)
  - 0 : False relation
NORMALIZED DISCOUNTED CUMULATIVE GAIN (NDCG)

➤ Used to evaluate the extracted pairs (ranked).

➤ The nDCG score takes into account if a good candidate is correctly ranked.

➤ The DCG and the iDCG formulas are almost similar except for the rank order, in effect the iDCG formula sorts in descending order.

➤ \( p \) is the number of relations extracted and \( \text{rel}_i \) corresponds to the score of the relation \( i \).

\[
\text{DCG}_p = \sum_{i=1}^{p} \frac{\text{rel}_i}{\log_2(i + 1)}
\]

\[
\text{nDCG}_p = \frac{\text{DCG}_p}{\text{iDCG}_p}
\]
PRECISION RESULT BETWEEN 4 METHODS

**Word Count**

<table>
<thead>
<tr>
<th>City</th>
<th>Genre</th>
<th>Nationality</th>
<th>US Politician</th>
<th>Capital</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.04</td>
<td>0.08</td>
<td>0.12</td>
<td>0.16</td>
</tr>
<tr>
<td>10</td>
<td>0.2</td>
<td>0.24</td>
<td>0.28</td>
<td>0.32</td>
</tr>
<tr>
<td>15</td>
<td>0.36</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

**Cosine Selection**

<table>
<thead>
<tr>
<th>City</th>
<th>Genre</th>
<th>Nationality</th>
<th>US Politician</th>
<th>Capital</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.04</td>
<td>0.08</td>
<td>0.12</td>
<td>0.16</td>
</tr>
<tr>
<td>10</td>
<td>0.2</td>
<td>0.24</td>
<td>0.28</td>
<td>0.32</td>
</tr>
<tr>
<td>15</td>
<td>0.36</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

**Euclidean Selection**

<table>
<thead>
<tr>
<th>City</th>
<th>Genre</th>
<th>Nationality</th>
<th>US Politician</th>
<th>Capital</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.04</td>
<td>0.08</td>
<td>0.12</td>
<td>0.16</td>
</tr>
<tr>
<td>10</td>
<td>0.2</td>
<td>0.24</td>
<td>0.28</td>
<td>0.32</td>
</tr>
<tr>
<td>15</td>
<td>0.36</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

**K-Means Selection**

<table>
<thead>
<tr>
<th>City</th>
<th>Genre</th>
<th>Nationality</th>
<th>US Politician</th>
<th>Capital</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.04</td>
<td>0.08</td>
<td>0.12</td>
<td>0.16</td>
</tr>
<tr>
<td>10</td>
<td>0.2</td>
<td>0.24</td>
<td>0.28</td>
<td>0.32</td>
</tr>
<tr>
<td>15</td>
<td>0.36</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>
NDCG RESULT BETWEEN 4 METHODS

Word Count

Cosine Selection

Euclidean Selection

K-Means Selection

City  Genre  Nationality  US Politician  Capital

NDCG

Number of Pairs in Input

NDCG

Number of Pairs in Input

NDCG

Number of Pairs in Input

NDCG

Number of Pairs in Input
RESULT OF INCREASING THE NUMBER OF PAIRS RETURN

➤ For the precision we obtain almost the same « shape » with lower scores when the number of returned pairs increases.

➤ For the NDCG score the result is the same, so it is not affected by the number of pairs.
We compare the results with the original implementation. Using fewer pairs in input (5 pairs instead of 20) can considerably reduce the execution time.
CONCLUSION

➤ This new implementation in distributed environment improves the computation time

➤ The corpus and the pairs chosen in input influence the extracted pairs.

➤ Input pairs selection methods improve the precision of the model with less pair in input.

➤ Evaluation can be done automatically.

➤ Word2Vec is very powerful for Relation Extraction.
FUTURE WORK

➤ Try our algorithms of « pair extraction » and our « input pair selection method » with other words embedding algorithms like GloVe from the Stanford NLP Group.

➤ One improvement can be to link the extracted information to a knowledge base of the type of relation, before the generation of similar words in the relation extraction part.
Thanks to Alisa Smirnova and for having supervised my thesis

and also thanks to the eXascale Infolab team.
END