Distributed System Programming - Final Report

Gil Shelef 305285470, Tal Wanish 302963541

August 9, 2016

Contents

1 Design and Flow

The general flow of our application is conducted as follows:

1. Find useful lexico-syntactic patters:
   (a) For each pair of words $w_1, w_2 \in Corpus$, extract the path between w_1, w_2 in the given sentence.
   (b) Save all the paths found in the previous step and appeared between at least $DPmin$ pairs.
   (c) Order the paths $p_1, p_2, \ldots, p_k$ in some order.
2. Build features vectors:
   (a) For each pair of noun words $w_1, w_2 \in Corpus$, build feature vector $v(w_1, w_2) := (v_1, v_2, \ldots, v_k)$, where v_i is the amount of sentences in which the path between w_1 and w_2 is p_i.
3. Train:
   (a) Based on pairs in $AnnotatedSet$ and their features vectors, train the classifier.

In the next sections we will describe the system main components and the "Map-Reduce" steps employed.

1.1 Classes

These are the principle classes that form the represention of the problem:

- $Word$ - this class stands for a word unit in the corpus.
  - word: the actual word as appeared in the corpus.
  - part of speech: the tag that representes the part of speech of the word in a specific sentence.
  - sentence position: the index of the word in the given sentence.
- $NounPair$ - represents a pair of noun words that appeared together in some senetence, along with their type: $true$, $false$ or $unknown$, according to the relation between the words.
- $DependencyPath$ - this class reperestens a path between a noun pair, e.g, if $NP_x\ w_1\ w_2\ \ldots\ w_k\ NP_Y$ is a path between NP_x, NP_y in some sentence, then $w_1\ w_2\ \ldots\ w_k$ is their $DependencyPath$.
- $Subsentence$ - this class consists of $NounPair$ and $DependencyPath$ that together form a subsentence.

1.2 Map Reduce Steps

Now we can detail each step of the "Map-Reduce" stage:

1. Step one:
   (a) *map*: given a $ngram$, the mapper parses it and emits a tuple of $\langle dp, (w_1, w_2)\rangle$ where $dp \in DependecyPath, (w_1, w_2) \in NounPair$ and dp is a non-empty path between w_1, w_2.
   (b) *reduce*: under a given key dp of type $DependencyPath$, the reducer receives all the pairs that are conneted by the path dp . The reducer checks if the path dp has at least $DPmin$ values of pairs and if indeed so, the reducer emits the path.

   After the first step we have our features list.

2. Step two:

   (a) *map*: based of on the results of the first step, the mapper scans the corpus again and for every pair $(w_1, w_2) \in NounPair$ who appeared in a path p_i found at the first step, the mapper emits the value 1 under the key $\langle (w_1, w_2), i \rangle$. This key represents the i'th coordinate in the feature vector of (w_1, w_2).

       Here we used a combiner (identical to the reducer) to locally sum up the intermediate values of the coordinates.

   (b) *reduce*: with the help of *compareTo* method that lexicographically compares the keys $\langle (w_1, w_2), i \rangle$, the reducer will recieve all the coordinates $i_1, i_2, \ldots, i_l$ for a given pair (w_1, w_2) in a row, where p_{i_k} is a feature (e.g, a path) that connects w_1, w_2 in some sentence.

After these two steps we have the features vector of every connected words in the corpus.

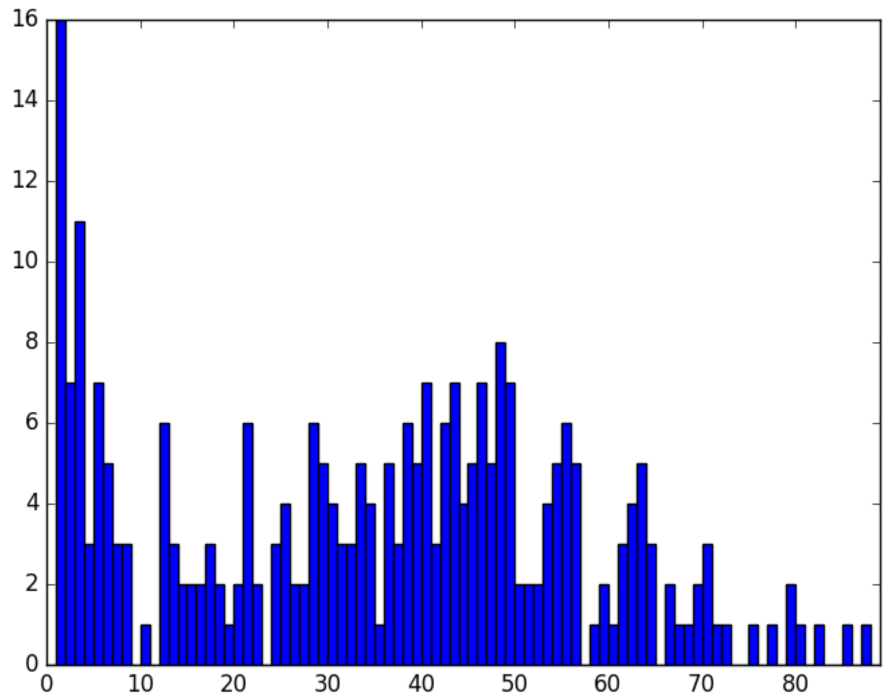
2 Analysis

Notation: For a given path dp we define $W(dp) = \left\{ (w_1, w_2) \in Corpus \cap AnnotatedSet \middle| \exists w_1 \cdot dp \cdot w_2 \in Corpus \right\}$. $W(dp)$ contains all the pairs of words w_1, w_2 that dp is a path between them for some observed sentence in the corpus. We also denote $w(dp) = |W(dp)|$.

When we tested our application with $DPmin = 5$ we got low precision rate. We suspected that the reason behind this is the amount of features: the meaning of low precision rate is answering "true" more then necessary. Thus, enlarging $DPmin$ value will reduce the number of features and therefore the chance of wrongly answering "true" is diminished.

In order to achieve better understanding of the possible values of $DPmin$, we ran a "Map-Reduce" cycle that counts $w(dp)$ for each dp. We conjuctured that greater values of $DPmin$ will result in smaller amount of features, but we wished to find an exact value of $w(dp)$ that will effectively filter some of the shared paths.

The histogram below shows the amount of paths that has the same $w(dp)$ value. As can be seen, most of the paths are around $w(dp) \approx 40$ and the average value of $w(dp)$ is roughly 46.



2.1 Results

In light of $w\,(dp)$ values, we decided to run the application with relatively large range. The table bellow details the results for each tested value of $DPmin$:

$DPmin$	Features	TP	FP	TN	FN	Recall	Precision	F-measure
5	329	266	150	0	0	1.0	0.639423076923	0.780058651026
10	321	259	107	0	0	1.0	0.707650273224	0.8288
15	314	273	133	0	0	1.0	0.672413793103	0.80412371134
20	306	287	127	0	0	1.0	0.693236714976	0.818830242511
25	298	272	131	0	0	1.0	0.674937965261	0.805925925926
30	297	271	132	0	0	1.0	0.672456575682	0.804154302671
35	294	285	129	0	0	1.0	0.688405797101	0.815450643777
40	292	273	142	0	0	1.0	0.657831325301	0.793604651163
45	289	280	141	0	0	1.0	0.665083135392	0.798858773181
50	285	270	146	0	0	1.0	0.649038461538	0.787172011662
55	283	274	143	0	0	1.0	0.657074340528	0.793053545586
60	281	285	136	0	0	1.0	0.676959619952	0.807365439093
65	280	273	133	0	0	1.0	0.672413793103	0.80412371134
70	280	270	148	0	0	1.0	0.645933014354	0.78488372093
75	278	281	126	0	0	1.0	0.690417690418	0.816860465116
80	276	279	141	0	0	1.0	0.664285714286	0.798283261803
85	**272**	**283**	**107**	**0**	**0**	**1.0**	**0.725641025641**	**0.841010401189**
90	271	273	123	0	0	1.0	0.689393939394	0.816143497758
95	270	279	127	0	0	1.0	0.687192118227	0.814598540146

As can be seen, the highest values of F-measure and precision are obtained with $DPmin = 85$.

When we analyzed few examples we observed that each “fp” pair has non-zero values for some features that appear between “tp” pairs. Moreover, when we examined which features are shared between “fp” pairs and “tp” pairs, we saw that most of the features are composed of one common word (e.g , “in”, “of”, “is” etc). Since those words are likely to appear between any pair of words, these features decrease the precision of our classifier.

2.1.1 FP Pairs

The table bellow shows some of the “fp” pairs and their related stemmed features:

FP Pair	Features
action chamber	taken by taken, of, by, in
empir prussia	in
power hitler	of
fibril heart	of
export valu	by
accumul marsh	in
throne russia	of
studi theatr	in, of
woman right	in
access support	of, to mechan, for, in

A possible solution for this problem may be supplying weights for the features according to their length. This idea is based on the intuitive assumption that longer features contain more information about the relation between the words and therefore could be more indicative of *hypernym/hyponym*.

2.1.2 TP Pairs

TP Pair	Features
absorpt activ	of, by, in, establish of
abus exercis	of, in
acceleromet devic	from
proport construct	of
regard attitud	for
accord write	with principl of
acronym word	of, for, are
adam men	are, rest of, of, among
account valu	from, of, as, have, for, in, of diffe between, for of, base, is
control condit	over

3 Communication

- First Step : $4,544,168$ keys were sent from the mapper to the reducer in total size of $207,380,812$ bytes.
- Second Step : 4544168 keys were sent from the mapper to the combiner in total size of $207,380,812$ bytes, and $482,994$ keys were sent from the combiner to the reducer in the approximated size of $25,598,682$ bytes.

4 Links

All the source code of the project including this document can be found here.