Unsupervised Language Learning - Miniproject
Replicating results of Dennis '05

Peter Csiba, petherz@gmail.com, UvANetId: 10506268
Peter Schmidt, petersch.home@gmail.com, UvANetId: 10506276

4th of April, 2013

Contents

1 Introduction

1.1 Problem to solve

Given a sentence with POS-tags induce the parse tree. We use exemplars from gold parses and induce only binary parse trees. As gold parses could have greater branching factor, best possible result is 88,1% computed by Klein and Manning[1].

1.2 How we divided our work

Peter Schmidt implemented the SNED algorithm, extracted topics with MALLET and experimented with their usage. Peter Csiba implemented the LSH algorithm, extracted the context cost function and implemented functions for working with parse trees. Both of us had long discussions about how we should approach the implementation of Dennis '05 article. We discuss that in the *Discussion* section. Also both of us made some experimentation (as normalization of cost function) and code run optimalization (we achieved 10-20 times better performance at the end).

2 Literature overview

2.1 Span-based normalized edit distance

The *edit distance* is a string metric, designed to measure the difference of two strings.

Let Σ be a finite alphabet, λ will denote the empty word, let $\Sigma_\lambda = \Sigma \cup \{\lambda\}$. By *elementary operation* we will understand ordered pair (a, b), where $a, b \in \Sigma \cup \{\lambda\}$, $(a, b) \neq (\lambda, \lambda)$. Such operation represents substitution of symbol a for symbol b. When defined this way, deletion (a, λ) and insertion (λ, b) can be considered special instances of substitution.

Elementary operations are weighted by a cost functon, an arbitrary function $\gamma : (\Sigma_\lambda, \Sigma_\lambda) \to R$ such that $\gamma(a, b) \geq 0$ for all a, b. Let $S = S_1 S_2 \dots S_n$ be a sequence of elementary substitutions, the cost function γ can be extended by letting $\gamma(S) = \sum_{i=1}^{n} S_i$.

The minimal edit distance of two character sequences, x and y can then be defined as

$$d(x, y) = \min_{S(x)=y} \gamma(S).$$

Furthermore, let $L(S) = L(S = S_1 S_2 \dots S_n) = n$ be the length of transformation S. Normalized edit distance of two character sequences, x and y is then defined as

$$d(x, y) = \min_{S(x)=y} \frac{\gamma(S)}{L(S)}.$$

[1] As we divide the WSJ-10 corpus to train and test corpuses then the number 88,1% is only approximate in our case.

Normalized edit distance generally computes an average cost per one symbol of the input character sequence. This enables it to be compared among strings of different lengths. Normalized edit distance is however not a metric any more. It is also worth mentioning, that the minimal normalized edit distance is not always the same, as post-normalized minimal edit distance. It can, hovewer, be computed in $O(nm^2)$ time, $n = |x|$, $m = |y|$, by an algorithm proposed by Marzal and Vidal.

To obtain the span based edit distance, the elementary operation (a, b) restrictions are relaxed to allow a, b to be words from the set Σ^*. When using such approach, a natural question would be how the cost function is supposed to look. Proposed approaches and our implementations will be discussed later.

Normalized span based edit distance is then defined in an analogous fashion and can be computed in $O(n^2m^3)$ time.

2.2 Topics model

The *topic model* is a probabilistic model of language generation. It views doxuments as mixture distributions of topics, while in turn topics are defined as mixture distribution of words. A Markov chain Monte Carlo method can then be used to estimate parameters of the distributions from given corpus.

In our approach, we extracted the documents for Topics model from Wikipedia. We used the implementation of Topics model provided by the machine learning toolkit MALLET, developed on the University of Massachusetts, Amherst. Our Wikipedia dictionary contained all in all 22649 words in 8138 documents, while the WSJ-10 corpus consisted of 6559 words. As some WSJ words were quite unique – most of them being numbers and company names – the Wikipedia articles matched $4337/6559 = 66.12\%$ of the WSJ corpus words.

2.3 Locality sensitive hashing

The generalized problem of *nearest neighbor search (NNS)* is states as follows: Given a set P of n objects represented as points in a normed space l_p^d, preprocess P so as to efficiently answer queries by finding the point in P closest to a querypoint q[2]. We then generalize for k nearest neighbors, which is a hard problem, so we relax the condition to found $(1 + \epsilon)$ solution in comparison with the best.

Idea of Locality sensitive hashing (LSH) is to hash points close to each other in l_1^d space so that they will collide with high probability. Having several of these hash functions makes buckets which could be searched in overall $O(k)$ time as it is enough if two points collide in only one hash value. Dennis is inspired by LSH to use similar method when searching for k-

[2]We use l_p^d to denote the Euclidean space $\mathbb{R}^d$ under the l_p norm, i.e., when the length of a vector $(x_1, \ldots, x_d)$ is defined as $(|x_1|^p + \cdots + |x_d|^p)^{\frac{1}{p}}$. Most of the time is enough to use the linear case of $p = 1$.

NN of a span. He gets nearest neighbors which are then aligned with SNED[3]. The motivation comes from computation cost as running SNED between sentences of whole corpus takes $O(ns^5)$, where n is the corpus size and s is the longest sentence length, instead of $O(ks^5)$ when using preprocessing with LSH. Note that Dennis uses hash functions which are lists of grammar rules which simplify spans and collisions are searched between these simplified spans.

We replicated this behaviour and for 267 of 519 (51.4%) sentences LSH finds less than 10 collisions, and for 151 of 519 (29.1%) finds no collisions at all. Therefore, when using LSH we use also bruteforce method for getting NN for sentences with less than one NN. Mostly, the sentences with no collisions are the longest ones.

2.4 Dennis '05 overview

After several readings and long thoughs about the article we are still not sure how it is actually implemented by Dennis. Even between us is some disagreement.

```
foreach sentence S in Corpus
  CS := HF collisions with S
  if size(CS) = 0
    then result := right branching
  else
    NN := k-nearest-neighbors from CS computed by SNED
    ConNN := all constituents from alignments of NN with S
    foreach possible binary tree parse BT of S
      ConBT := all constituents of BT
    result := BT with max size intersection of ConNN and ConBT
```

Discutable parts.

- Use SNED for getting k-NN. We use that and we think Dennis uses only LSH to get nearest neighbors. g
- Use SNED to get alignments of sentences to derive constituents rather than using gold parses of NN to derive candidate constituents. We use gold parses and that is not so unsupervised as alignments which Peter Csiba thinks Dennis uses.

[3]Maybe SNED is used to make the set returned by LSH even smaller, we are not sure.

- Using Topics model for computing the cost function of span replacements. Dennis states only the dot product of topic distributions of contexts:

$$\gamma(span_1, span_2) = \sum_{t \in topics} P(t|span_1)P(t|span_2).$$

We assume that this dot product is computed for both left and right contexts of the given spans. Another problem is that SNED uses costs and not probabilities. So we decided to transform it $\gamma'(s_1, s_2) = \frac{1}{\epsilon+\gamma(s_1,s_2)}$ and then normalize it to the interval $[0.5, 1.0]$. We also tried $\gamma'(s_1, s_2) = 1 - \gamma(s_1, s_2)$ and other types of normalization but all gived slightly inferior results.

3 Our model

3.1 Overview

After approximately understanding the Dennis '05 article we concluded that it will be hard to implement it in its full size. So we decided to make a simple first version and then iteratively expand it.

```
Divide corpus with ratio 90/10 to train TraC and test TesC
CCF := cost function for each two spans occuring in same
   contexts
foreach sentence S in TraC
  NN := get k-nearest-neighbors from CS computed by SNED
  ConNN := all constituents of gold parses of all NN
  foreach possible binary tree parse BT of S
    ConBT := all constituents of BT
  BesBT := BT with max size intersection of ConNN and ConBT
```

Cost function. From the train corpus we computed the probabilities $\gamma(span_1, span_2) = \sum_{c \in contexts} P(c|span_1)P(c|span_2)$. We extracted 38796 spans and computed 6491349 values of γ - other were implicitly zero as such two spans never occured in a same context[4]. As SNED works with costs and not probabilities we transformed non-zero probabilities to costs $\gamma'(s_1, s_2) = \frac{1}{\gamma(s_1,s_2)}$. As SNED needs cost for all pairs of spans we set the cost for spans not occurying in a same context, i.e. $\gamma(span_1, span_2) = 0$ to $\frac{length(span_1)+length(span_2)}{2}$ after some experimentation.

[4]So $1 - \frac{6491349}{38796^2} = 99.56\%$ were zero valued and these values could be implicitly stored without need of additional memory and thus made the approach of precomputing cost function values feasible.

Extended contexts with topics. After discussion we implemented the derived topics to cost function by counting each span in context $(word_1, word_2)$ as in all contexts (t_1, t_2) where t_i is one of the 25 closest words to word w_i, i.e. having greatest dot product of topic distribution. For example for word *born* the closest t_i are: "died, where, career, worked, father, life, years, married, became, wife, moved, him, ..." . For example for span (NN, NN, VBD) in context $(the, born)$ we added (NN, NN, VBD) the following contexts: $(the, born), (of, born), (and, born), (the, died), (the, where), (of, died), (and, died), (of, where), \ldots$. This lead to 21643306 entries (3 times more) in cost function. Because of our operational memory restrictions we used only third of the most probable entries.

3.2 Algorithm

```
Divide corpus with ratio 90/10 to train TraC and test TesC
CCF := cost function for each two spans occuring in same
   contexts
TP := topic probabilities with word probabilities from
   Wikipedia articles
HF := hash functions as permutations of most used grammar rules
    in TraC

foreach sentence S in TraC
  CS := HF collisions with S
  if size(CS) < k
    CS := TraC
  NN := get k-nearest-neighbors from CS computed by SNED
  ConNN := all constituents of gold parses of all NN
  foreach possible binary tree parse BT of S
    ConBT := all constituents of BT
    BesBT := BT with max size intersection of ConNN and ConBT
```

3.3 Versions

We considered four versions of the algorithm above: with/without Topics and with/without LSH.

Without Topics. In this case instead of using *extended contexts with topics* (discussed in 3.1) we use only non-zero probabilities for spans occuring in the same context. See our first version at 3.1.

Without LSH. For each sentence we iterate through all exemplars from train corpus and compute its distance with SNED. Then we pick 30 closest and use them as nearest neighbors. That's about 10-20 slower as with LSH and the difference could be even greater for bigger sentences as LSH is $O(nk)$ and SNED is $O(nk^5)$ where k is the number of sentences.

4 Results

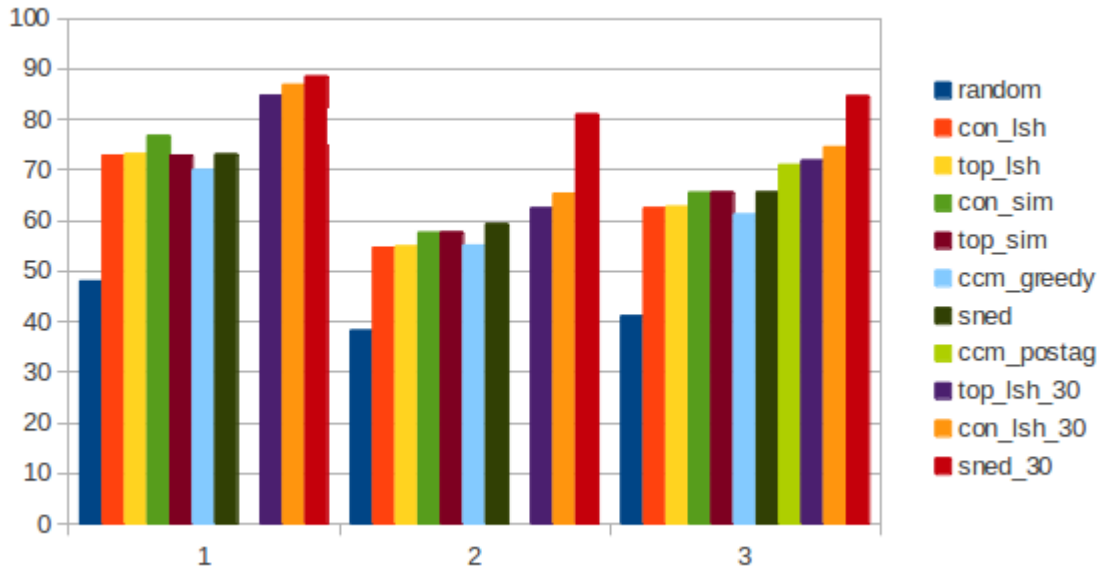
4.1 Evaluation

We use the same evaluation metric as Dennis '05. Let S be the parse derived with our algorithm and T be the target parse. Then we get all constituents $conS$ and $conT$ for S

and T expect the *trivial* ones, i.e. the whole sentence[5]. *Recall* is then the mean proportion of *conT* that the model proposed. *Precision* is the mean proportion of *conS* that appear in the gold standard. *F1 score* is the harmonic mean of recall and precision. Note that as same constituent could occur multiple times and thus make the unlabelled recall over 1.0 we compute labeled recall instead - we assume Dennis '05 does the same even if he mention unlabelled recall.

The *overall* recall, precision and F1 is computed from summing all constituents from all sentences derived (again, except trivial). So it is not the average of the individual F1 scores for the sentences.

4.2 F1 Scores



We see that from the four configurarions of with Topics/without Topics and with LSH/without LSH the with Topics without LSH achieved the best results. Moreover the results are almost the same as results of Dennis '05 and we are really happy for that.

We were surprised that "with Topics" gived us only a very small improvement. Still, it should be considered as the algorithm is deterministic, i.e. without any random decisions.

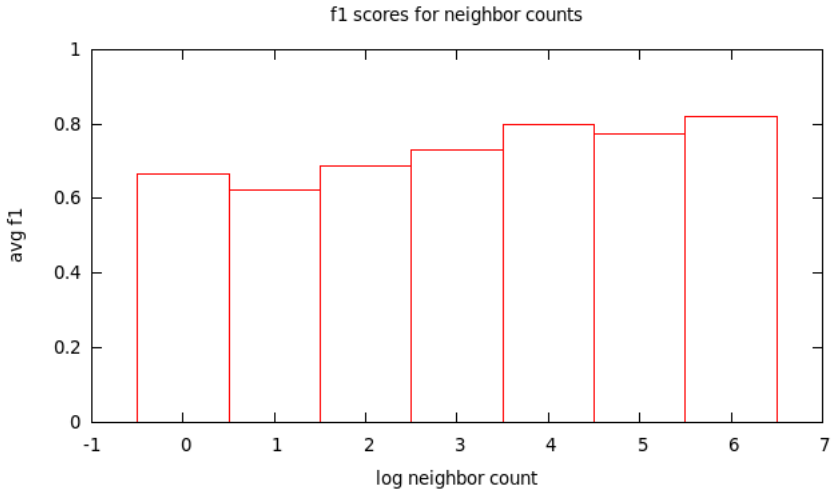
On the other hand, we were not able to reconstruct the SNED30 result (at least we achieved similar recall values). Peter Csiba guesses that this could be because of the parameters of the LSH function. Dennis uses 300 rules in 5 hash functions - we found this setting

[5]We think that Dennis '05 also don't counts them as the result are then considerably higher (about 4-6 points of F1 score).

inferior - so we use 600 rules and 10 hash functions and therefore the number of sentences satisfying the 30 NN limit is greater. In our case 192 of 519, i.e. 37% of sentences satisfied that condition.

4.3 LSH

On the following chart we see how the F1 values depend on the nearest neghbors found by LSH. Note that nearest neighbour count is on a logarithmic scale.



We guess that the LSH function clusters similar sentences in some way. Because they are similar, the exemplars are also more similar and the derived parses are therefore similar to the gold parses.

5 Discussion

5.1 Topics Model

Other method for implementing Topics model was to compute the context topic dot products directly when running the SNED algorithm. In this case all the contexts get a non-zero probability as the topics vector for each word is non-zero at all places. But this produced no good results as F1 score was only around 54. Moreover, it was also computationaly infeasible.

5.2 Performance

Using the profiler tool and some cacheing we were able to tweak the running time. For the LSH version it run in 5 minutes on the 519 test sentences what was about 40 times

faster then the first version. Without the LSH the running time was about 90 minutes on a standard laptop.

5.3 Conclusion

There is indeed a lot further work which could be considered. Most importantly try out using no gold parses and use only alignments provided by SNED so the algorithm will be completely unsupervised and could be run on the whole corpus without the need to divide it to train and test. Second idea is extracting topics directly from the whole WSJ corpus could be considered as then all the words will be present.

We both enjoyed the miniproject, especially the discussions about the interpretation of the article and founding feasible possible implementations. Also the coding was fun, expecially when the time and memory constrains come up. We are happy with the results.

6 Bibliography

- Griffiths & Steyvers (2002). Prediction and semantic association. In Nips.
- Dennis (2005). An Exemplar-based Approach to Unsupervised Parsing.
- Marzal & Vidal (1993). Computation of normalized edit distance and applications.
- Gionis, Indyky & Motwaniz (1998). Similarity Search in High Dimensions via Hashing.