

Some algorithms and computational techniques in NLP

1 Introduction

[TBW]

2 Tokenization

Tokenization or text-processing is the process for cleaning text and converting it into standard format:

- Remove redundant tokens, e.g. HTML tags.
- Finding word boundaries, using white-spaces and punctuation. One needs to be careful about different type of punctuation, for example *st.*, *B.Sc*, *km*, *miles*, ... could have different interpretation if they are considered separately.
- Stemming/Lemmatization: In many languages, one family of words might appear in different forms. For example in English, *go*, could be used in any of the following forms, *went*, *goes*, *gone*, *etc* depending the structure of the sentence. To make the input easier to be processed (at least from some specific models' point of view), one can replaces all such occurrences with *go*. The most important stemming is Porter Stemming. The SnowBall is also a very important language preprocessing package.
- Removing stop-words: Removing many of the words in the sentences which don't have much of semantic meaning, e.g. *of*, *that*, *the*, *a*, *an*, One such list could be found here .
- Removing capitalization: In many cases it would help to remove the the capitalization. One standard counterexample is *US vs. us*.

Many of the above steps, may or may not be needed, depending on the target problem.

3 Language models

3.1 Language as a stochastic process

Assume that words of a language is sampled from a Multinomial distribution,

4 tf.idf

The *tf.idf* is one of the features that could be used for modeling documents. The *tf* vectors are base on the relative word frequencies, i.e. the most frequent word has the most *tf* value:

$$tf_w = \frac{c(w, d)}{\max_v c(v, d)}$$

where $c(w, d)$ is the word count in document d . The *idf* is aiming at eliminating stop-words which appear in most of the documents, like “the”.

$$idf_w = \log \frac{N}{n_w}$$

where n_w is the number of the documents that w appears in. If the word appears in almost all of the documents, we have $n_w \approx N$, then $idf \approx 0$. The *tf.idf* score of a word is:

$$tf.idf(w) = tf_w \times idf_w$$

5 Parsing

5.1 Inside-Outside algorithm

Inside-Outside could be considered as generalization of forward-backward algorithm in HMMs, for model probability estimation in probabilistic context-free grammars. In context-free grammar we only have the rules of only the following form:

$$i \rightarrow jk \text{ and } i \rightarrow w$$

in which i, j, k are integers which correspond to unique internal nodes. In the Probabilistic Context-Free Grammar, we describe each of these rules via some probability distributions:

$$\mathbb{P}(i \rightarrow jk) \text{ and } \mathbb{P}(i \rightarrow w)$$

with the proper probability distribution on each of the rules:

$$\sum_{i,j,k} \mathbb{P}(i \rightarrow jk) + \sum_i \mathbb{P}(i \rightarrow w) = 1$$

Similar to HMMs we ask the following three important questions:

- Given grammar \mathcal{G} , what is the probability of a sentence, i.e. $\mathbb{P}(w_{1:m}|\mathcal{G})$?
- Given a grammar and a sentence, what is the most-likely parse tree, i.e. $\arg \max_t \mathbb{P}(t|\mathcal{G}, w_{1:m})$?
- Given a sentence, what is the grammar that maximizes its probability of observation, i.e. $\arg \max_{\mathcal{G}} \mathbb{P}(w_{1:m}|\mathcal{G})$?

To define the above probabilities, we define the following two probability distributions, to make the derivation and its interpretation easier. Here the notation, and partly the representation is from [?]. A good reference for this algorithm is [?].

Definition 1 (Inside and Outside probabilities).

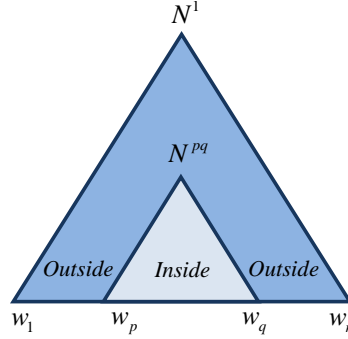
Inside probability: $\beta_j(p, q)$, ($p \leq q$) is the probability of generating $w_p \dots w_q$, given the nonterminal node N_j , i.e. $\mathbb{P}(w_{p:q}|N_{pq}^j, \mathcal{G})$.

Outside probability: $\alpha_j(p, q)$, ($p \leq q$) is the probability of generating $w_1 \dots w_{p-1}$ and $w_{q+1} \dots w_n$, beginning with the nonterminal node N_1 and generating the nonterminal node N_{pq}^j , i.e. $\mathbb{P}(w_{1:p-1}, N_{pq}^j, w_{q+1:n}|\mathcal{G})$

It can be shown that we have the following recursive formulas for the inside and outside formula:

$$\beta_j(p, q) = \sum_{r,s} \sum_{i=p}^{q-1} \mathbb{P}(N^j \rightarrow N^r N^s) \beta_r(p, i) \beta_s(i+1, q)$$

$$\alpha_j(p, q) = \sum_{r,s \neq j} \sum_{i=q+1}^n \alpha_r(p, i) \mathbb{P}(N^r \rightarrow N^j N^s) \beta_s(q+1, i) + \sum_{r,s} \sum_{i=1}^{p-1} \alpha_r(i, q) \mathbb{P}(N^r \rightarrow N^s N^j) \beta_s(i, p-1)$$



5.1.1 The probability a specific sentence, given the grammar

The probability of observing one specific sentence is the following:

$$\mathbb{P}(w_{1:n}|\mathcal{G}) = \sum_j \alpha_j(p, q) \beta_j(p, q)$$

5.1.2 The max-likely parse tree

Define the following variable:

$\delta_i(p, q) :=$ Max probability of the parse tree, that spans p to q , rooted at the internal node N^i

We can expand this variable recursively in the following way:

$$\delta_i(p, q) = \begin{cases} \max_{1 \leq j, k \leq n, p \leq r < q} \mathbb{P}(N^i \rightarrow N^j N^k) \delta_j(p, r) \delta_k(r + 1, q) \\ \mathbb{P}(N^i \rightarrow w_p) \end{cases} \quad \text{if } p = q$$

Using the above recursive definition and memoization of the values for $\delta_j(\cdot)$ we can calculate the value of $\delta_1(1, n)$, which contains the value of the best parse tree for the whole sentence, in time $O(n^3 g^3)$. If we save the decisions while memoizations, we can backtrack the decisions and recover the optimal tree.

6 N-grams and discounting

Write from <http://nlp.stanford.edu/wcmac/papers/20050421-smoothing-tutorial.pdf>

References

- [1] Karim Lari and Steve J Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56, 1990.
- [2] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.