Jožef Stefan International Postgraduate School

Seminar work in New Media and Language Technologies

CONSTRUCTING N-WAY ALIGNMENTS USING MULTIPLE PAIR-WISE ALIGNMENTS

Seminar mentor:
Prof. Dr. Tomaž Erjavec

Student:
Darko Čerepnalkoski

Research advisor:
Prof. Dr. Sašo Džeroski

Ljubljana
2008

1. Informal Introduction

The goal of this seminar work is to describe an algorithm for building N-way sentence alignments from existing pair-wise sentence alignments. More specifically, when given **n** pair-wise alignments, such that one of the corpora in the alignment is the same for all of the alignments, and the other is different for all of the alignments – it should produce **n+1**-way alignment. The corpus which is included in all of the alignments is called the **hub corpus**.

Example:
Given three pair-wise alignments: EN-MK, EN-SI and EN-DE, it will produce one 4-way alignment EN-MK-SI-DE (Figure 1).

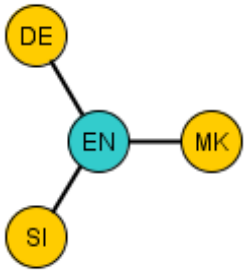

Figure 1.

The definition can be generalized to include any alignment and not just pair-wise alignment. So, when given set of alignments of any cardinality, such that there exists only one hub corpus and all other corpora participate in only one alignment, the algorithm produces an alignment which includes all the corpora.

2. Definitions

In this sections are given formal definition of some concepts that are needed for the description of the algorithm.

Corpus is a non-empty set of sentences: **C** = $\{s_1, s_2, \ldots, s_n\}$, $n \geq 1$
Corpus Set is a non-empty set of corpora: **CS** = $\{C_1, C_2, \ldots, C_c\}$, $c = |CS|$ $c \geq 2$

Alignment A has the characteristic (profile) **α(A)**=$(C_{i1}, C_{i2}, \ldots, C_{i\beta})$, $\beta = \beta(A) \geq 2$ is cardinality of A. The profile tells which corpora this alignment aligns, and in which order, all C_i are different.
Alignment A is a non-empty set of links: **A** = $\{l_1, l_2, \ldots, l_n\}$, $n \geq 1$
Link is a tuple of link groups: **l** = $(lg_1, lg_2, \ldots, lg_\beta)$, where $lg_j \subseteq C_{ij}$. (A linkgroup is a set of sentences from a corpus).
If $l_1 = (lg_{11}, lg_{12}, \ldots, lg_{1\beta})$ and $l_2 = (lg_{21}, lg_{22}, \ldots, lg_{2\beta}) \in A$ are two links, then $lg_{1i} \cap lg_{2i} = \varnothing$. In one alignment a sentence can appear in at most 1 linkgroup in one link. Which linkgroup it can be in is determined by the profile of the alignment.

Alignment Set is a non-empty set of alignments: **AS** = $\{A_1, A_2, \ldots, A_a\}$, $a = |AS|$ $a \geq 1$

3. Algorithm

The algorithm proceeds in two phases. In the first phase a data structure is built. This data structure maps sentences to sets of links. In the second phase, the algorithm for building the result alignment is run.

Phase 1:
The mapping SL is built.

SL – a mapping: sentence **S** → set of links {**l** | **S** appears in **l**}. For each sentence S of any corpus C, it gives the set of links in which S appears. For every alignment A, there can be at most one link in which S appears. So, if the number of alignments is *a*, there can be at most *a* such links total.

default: SL(S) = ∅, for all sentences S

for each A in AS
  for each l=(lg_1, lg_2, …,lg_n) in A
    for i=1 to n
      for each S in lg_i
        SL(S) ← SL(S) ∪ l

Phase 2:
The resulting alignment A is built.

Input:
CS = {C_1, C_2, …, C_c} - corpora to align
AS = {A_1, A_2, …, A_a} - alignments that are used to align the corpora. The alignments are such that: ∃! C_h ∈ CS such that appears in the profile of every alignment – called the hub corpus. Every other corpus appears in only one alignment.
SL – the mapping build in Phase 1.

Output:
A' – the wanted result alignment of the alignment of CS using AS

Additional structures:
LC – a mapping: link group lg → corpus C. Let l = (lg_1, lg_2, …, lg_n) ∈ A. And α(A) = (C_1, C_2, …, C_n). Then LC(lg_i)=C_i.
TS = ∅ - Traversed sentences set – initially empty
TL=∅- Traversed links set – initially empty
SQ = *empty queue* - Sentence queue (a queue of sentences) – initially empty
C_h – hub corpus

for each S_c in C_h
  if $S_c \notin$ TS
    SQ.enqueue(Sc)

```
l' ← (lg1', lg2', …, lgc'); new link, where lgi' = ∅
while SQ is not empty
    S←SQ.dequeue();
    TS ← TS ∪ S
    for each l in SL(S)
        if l ∉ TL
            TL ← TL ∪ l
            for each lgi in l
                lgj' ← lgj' ∪ lgi, where lgj' is such that LC(lgj')=LC(lgi)
            lg = linkgroup of l such that LC(lg) = Ch
            for each s in lg
                if s ∉ TS then SQ.enqueue(s)
```

4. Performance Considerations

The time complexity of the algorithm is O(S*A*C), where S is the number of sentences in the hub corpus, A is the number of input alignments used and C is the average number of corpora per alignment. In the standard scenario of using only pair-wise alignments C = 2.
For the main algorithm two data structures are important: the set of alignments and the mapping SL. The space complexity of the data structure for the alignments is O(A*C*S), where A is the number of alignments, C is the average number of corpora per alignment and S is the average number of sentences per corpus.
The space complexity of the data structure for the mapping SL is O(C*S), where C is the total number of corpora in the result alignment and S is the average number of sentences per corpus.