PERFORMANCE ANALYSIS OF TEXT ALIGNMENT TOOLS USING AUTOMATED TEST BED

John Nabil[1] and Mohamed Waleed Fakhr[2]

[1] Arab Academy for Science, Technology & Maritime Transport, Cairo, Egypt
johnnabil@gmail.com
[2] Arab Academy for Science, Technology & Maritime Transport, Cairo, Egypt
waleedfakhr@yahoo.com

ABSTRACT

The text alignment process is an important activity in the statistical machine translation (SMT) upon the training phase of SMT systems on huge amounts of parallel corpora. We are evaluating five of the popular text aligners and analyzing the methodologies used in each of them, then we are introducing new open source automated test bed for testing those text aligners and generating reports called pyTester, and finally discuss our observations about the test results generated from pyTester.

KEYWORDS

Arabic, Automatic text alignment, English, French, parallel corpora, statistical machine translation, SMT

1. INTRODUCTION

Text alignment is a process that takes two text streams -the first one is written in one language, and the other text stream is the same text translated in another language – and converts them to aligned text

Aligned Text is done by marking the beginning and end of a chunk of text stream in one language with the beginning and end of the aligned chunk in the other stream of the second language

Here is simple illustration for the statistical machine translation as noisy channel model:

- Statistical MT is based on the noisy channel model
- Developed by Claude Shannon to model communication , 1948 [9](e.g., over a phone line)

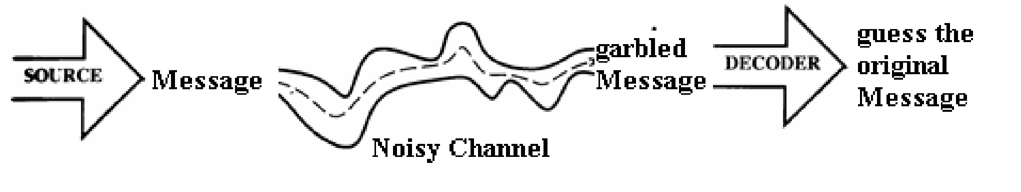

Figure 1: Illustration for the noisy channel concept

- Noisy channel model in SMT (ex. E $\rightarrow$ A):

- Assume that the source text is in English
- But when it is transmitted over the noisy channel, it somehow gets garbled and comes out in Arabic
    - i.e. the noisy channel has garbled the original English input into Arabic
- Arabic may be regarded as form of garbled English.
- In Translation we have the input and we want to get the garbled message

We are going to discuss the importance of text alignment in SMT in section 2, then mention the most popular text alignment methodologies in section 3, and analyze the most the important text aligners in section 4, and in section 5, we introduce the new automated test bed which is pyTester and in section 6, we show the results of running pyTester with the five text aligners, and finally we give conclusion about our work.

2. TEXT ALIGNMENT SIGNIFICANCE

In Statistical Machine Translation we need very huge chunks of aligned text, especially when we do translation between more than one pair of languages, and the process of manual alignment for that huge chucks takes a lot of time, for example if we assume that one person can align 500 word of Arabic text with the corresponding translated text of English text, in one hour, so aligning 1 million of words of Arabic text requires 2,000 man hour, but in reality we may want more than one billion of words to from many sources to have balanced corpus. Which can take 2-million man-hour, and if we assume that the human aligner works 8 hours/day, so it needs 250,000 working days, you can imagine how much expenses you have to pay to just align 1 billion of words. Even 1 billion words may not be enough quantity of aligned corpus.

I did not calculate the time and expenses of collecting the text of multiple languages because it is very easy for any crawler to collect very large quantities of bilingual text. So the problem lies in aligning this quantity of text.

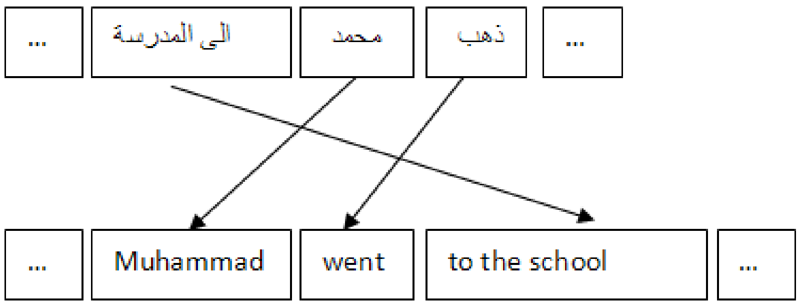

Figure 2: Illustration for word alignment between Arabic and English words

So we are targeting this problem by automating the process of aligning very large quantity of bilingual text to produce well-aligned large bilingual corpus.

2.1. What is the importance of automated text alignment?

We discussed above, how the automation of the text alignment can reduce the number of working hours.

We mentioned that 1 billion words needs 2 million working hour, if we assume the human aligner takes about 1 dollar per hour, then we need 2 million dollars, which of course exceeds the budget of many projects.
So the most important part of automating this process is to improve the economics of SMT.

2.2. English-Arabic text alignment situation: [8]

1) First, a single word in Arabic can have many variations depending on the morphological variations that it can undertake. Like English, Arabic adds prefixes and suffixes to a word to form other variants. However, unlike English, Arabic can also add infixes to words. This makes algorithms for English morphological analysis not applicable to Arabic
2) The prefixes and suffixes of a word may not always be attached to the other letters in the same word
3) In Arabic a whole English sentence can be represented using one single word (e.g. "and I met him" is translated to " وقابلته ")
4) Unlike English, the prepositions and pronouns are not separate words. Prepositions and pronouns in Arabic are usually attached to the word (for example, the 2-word English phrase "his book" is translated to a single word "كتابه " in Arabic)
5) A single English word can have a phrase Arabic translation (e.g. "غير مزود بالسلاح" is translated to "unarmed").
6) Spaces, in Arabic, might not separate two words from each other. For example, the conjunction "و" is usually written without a space between it and the next word
7) The word order in an Arabic sentence is usually different from that in the English sentence. In fact, the word order might change in different Arabic translations of the same English sentence
8) Arabic sentences might not contain any verbs.
9) Most of the text alignment tools were created for aligning Latin languages, which is different than Arabic in aligning because they have more punctuations, proper names, acronyms…
10) The linguistics information used in most of the alignment tools is very shallow.

3. SURVEY ON TEXT ALIGNMENT METHODOLOGIES

There are many methods used currently in Text alignment.

3.1 Statistical Methods

3.1.1. Gale and Church, 1993 [4]

It works on the principle that equivalent sentences should be roughly similar in length too, that is longer sentences in one language should correspond to longer sentences in the other language.
There is score based on probability is assigned to each pair of sentences aligned, this score is based on the scaled difference of lengths of the two sentence, then this score is used in a dynamic programming framework, to be able to find the maximum likelihood alignment of sentences.
The model used in this method, was inspired by the observation that longer regions of text most probably will have longer translations, Especially It was discovered that the correlation between the length of paragraph in terms of character count and the length of its translation is extremely high, So the correlation of the sentences lengths is very strong method for sentence alignment.

a. Based on character based sentence length correlations
b. Tools that follow this method: “Bilingual Sentence Aligner” (made by Microsoft)[7], Vanilla [2] and Hunalign [11]

3.1.2 The IBM Alignment Models 1 though 4

It Models the probabilistic relationship between the source language string *f* and the target language string *e* and the alignment *a* between positions in *f* and *e*.

This is brief illustration for the IBM alignment:

1) Model 1: lexical translation
   -Example:

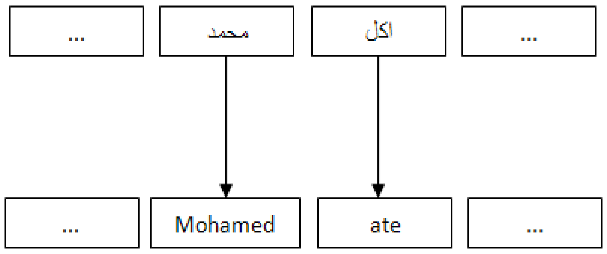

Figure 3: Example of using Model 1 in text alignment

2) Model 2: adds absolute reordering model
   -Example :

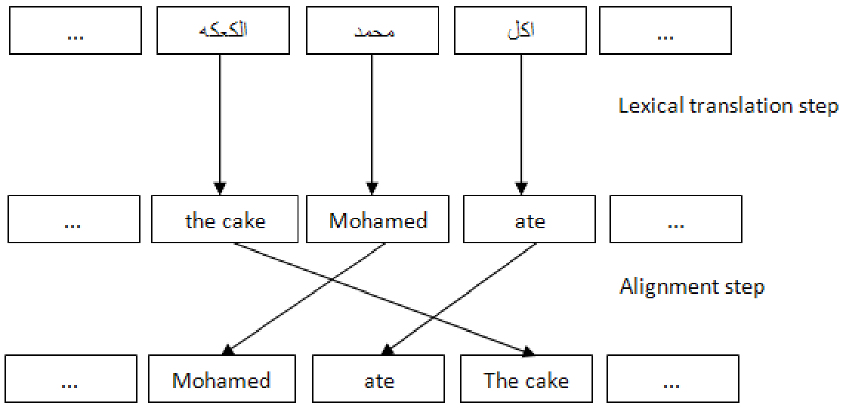

Figure 4: Example of Model 2 of text alignment

3) Model 3: adds fertility model (fertility of a word: means number of words in the target language that correspond to this word)

   - We use null word to model the word insertions
   - Example:

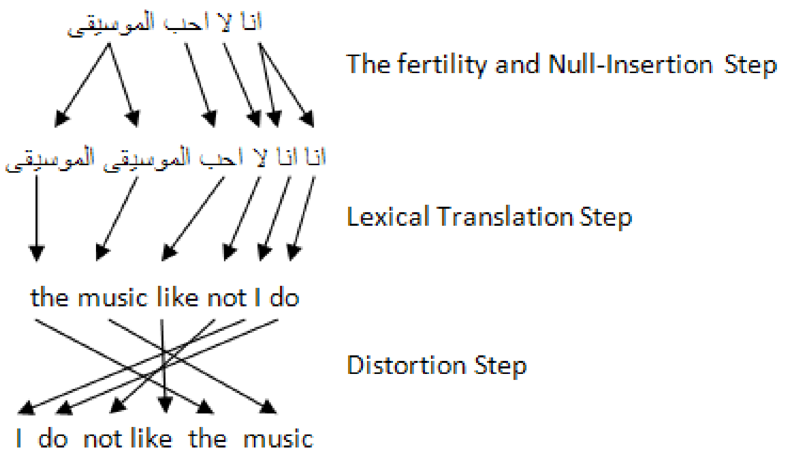

Figure 5: Example of Model 3 of text alignment

4) Model 4: relative reordering model
   - It uses the relative position instead of absolute position (Model 2)
   - It can be useful in long sentences.

One of the Tools that follow IBM models is Hunalign [11].

3.2. Heuristic Methods

The Heuristic methods differ from the statistical methods by being related to specific associative measures instead of statistical measures.

3.2.1 Dice coefficient association

a. It is one of the most basic heuristic word alignment techniques
b. Assign co-occurrence scores to each word pair and then it chooses the aligned words based on the highest co-occurrence score.
c. Tools that follow this method: Unplug [10]

3.2.2 The K-vec alignment algorithm, 1994 [3]

a. Used for dictionary extraction.
b. Uses several heuristic measures to estimate word correspondences, the heuristics are applied in sequence, i.e. heuristic A is used to make a first candidate selection, heuristic B is used to narrow down the selection made by heuristic A and so on.
c. Heuristic A and B may be language dependent heuristics, so they can be customized to the nature of the language pair.
d. The first stage alignment selection is made based on word occurrence vectors, the corpora to be analyzed is divided into K number of segments and for each word, a K-dimensional binary vector is constructed.
e. If the word appears in segment k, the corresponding dimension in the vector is set to 1, for a document divided into three segments, where the word "car" occurs in segment 1 and 1, the binary vector for car would be [1,0,1].
f. All source and target language words are assigned a k-vec and are later compared with each other. High correlation between segments where the word occurs and does not occur is

thought to raise the likelihood that a certain source word corresponds with a certain target word

3.2.3 Simple Hybrid Aligner, 1998 [1]

a. Combines the K-vec approach with a word to word search algorithm
b. This system is designed mainly to combine some simple assumptions about the translation process to extract alignment for words with low frequency. So that the algorithm was implemented in modular way to allow the user to test and analyze different combinations of techniques and assumptions about the translation process.
c. One of the disadvantages of this system is that it is trying to find word and phrase alignment for a bitext that is already aligned at the sentence level which is not always available at the early stages of the text alignments especially when we have some spurious sentences in one language that don't have correspondent sentences in the other language.
d. Tools that follow this method: Align

3.2.4 Geometric alignment, Melamed 1999[6]

a. Trying to find maps of correspondences in the bitext (TPC: true point of correspondence)
b. Its algorithm is called SIMR (Smooth Injective Map Recognizer)
c. Tools that follow this method: GMA

3.2.5 Clue-based alignment, 2003[10]

a. Uses a segment-to-segment matrix, a clue matrix, to represent the possible alignments, and assigns a score to each available word alignment.
b. The score that results from a weighted summarization of the independent clue sources, examples of such clue sources are the dice co-efficient,
c. Longest common subsequence ratio, POS tags, positional weighting, N-gram, and chunks (n-grams and chunks are used as clues for multi word unit discovery).
d. Tools that follow this method: UPlug [10]

4. POPULAR TEXT ALIGNMENT TOOLS

4.1. Bilingual Sentence Aligner:

- This is a alignment tool was created by Microsoft research center, it is implemented by PERL, and used for finding which sentences do translate one-for-one in a parallel bilingual corpus.
- The first approach shown to be effective at aligning large corpora was based on modeling the relationship between the lengths of sentences that are mutual translations.
- The length-based methods require no special knowledge about the languages.
- The word-correspondence-based methods of Chen and Melamed do not require this sort of information about the corpus, but they either require an initial bilingual lexicon or they depend on finding cognates[1] in the two languages to suggest word correspondences.
- Wu's method also requires that the bilingual lexicon be externally supplied.
- In this system, we find hybrid sentence-alignment method, using previous sentence-length-based and word correspondence-based models.
- Algorithm:

  - First: They employ a novel search pruning technique to efficiently find the sentence pairs that align with highest probability without the use of anchor points or larger previously aligned units
  - Second: they use the sentence pairs acquired from the first step to train IBM translation model 1.
  - Finally: they realign the corpus with the initial alignment with IBM model 1.

- According to their algorithm, it does not require supplied lexicon.
- For first step of the algorithm: they use the Poisson distribution, because when they find that the Poisson distribution fits the data better than Gaussian distribution.
- For second step of the algorithm: the search is done using the dynamic programming, and they avoid the problem of exhaustive search by assuming that the best alignment should be found within some boundaries according to heuristic function.
- Disadvantage: it aligns only the sentences that have 1-to-1 mapping between the target and source languages

4.2. GMA (i.e. Geometric Mapping and Alignment)

- The basic algorithm used here is called Smooth Injective Map Recognition (SIMR).
- Features of SIMR:

  - Allows points of correspondence to occur for word order differences
  - It's output can be converted easily into sentence alignment

- Most of alignment algorithms assume that there are text unit boundaries.
- How does SIMR work?

  - It searches for good points of correspondences in the parallel text.
  - First, generate number of correspondence points
  - Second, select the points that satisfy the true point of correspondence conditions

- Point Generation:

  - A point (x, y), where x is the position of token e in the source language, and y is the position of token f in the target language, is chosen if e and f are likely to be mutual translations.
  - Do we need very accurate lexicon?
    - No
  - Etymological Similarity:

- It is very useful to get the related words

  - The phonetic cognateness between words in different languages is used to map the words together.
  - Matching Predicates:

    - Longest common subsequence Ratio (LCSR) is the number of characters that appear in the same order in both tokens divided by the length of the longer token.
    - Stop-list of closed-classes

        - Using translation lexicon, it returns true if there's an entry for the token pair in the lexicon.

- Point Selection:
    - There are some conditions for the True Points of Correspondence (TPC):
    - Linearity: TPCs should be roughly linearly arranged, this is judged by the Root mean square distance.

- Constant Slope: the chain's slop should be similar to the bitext slope, this is done by defining maximum deviation angel threshold.
- Injectivity: no two points in the same chain have the same x co-ordinates or y co-ordinates.
- Reducing the Search Space:
    - They used the previous conditions for TPCs to reduce the search space of available TPC chains
- Disadvantage: Source and target language files should be in the axis format which is not convenient

4.3. Vanilla

- They are dealing with technical literature and official text with many "itemized" paragraphs.
- They assume that aligned sentences will have approximately the same length (length here is measured in terms of number of characters)
- Needs special format
- The code seems to be very old, because it used the very old C style in function declaration (http://software.intel.com/en-us/articles/old-style-c-function-argument-declarations-are-supported/)
- One of the disadvantages for this tool, if the two files (source and target languages) contain different number of paragraphs it returns an error (e.g. "align_regions: input files do not contain the same number of hard regions")

4.4. Hunalign

- They concentrated on the dictionary and length-based approaches
- Algorithm:
    - First: they generate a rough translation of the source text by using the dictionary
    - Second: Similarity score is calculated on two major components (token-based and length-based)

- Token-based similarity: number of shared words in the two sentences, normalized by the larger token count of the two sentences.
- Length-based similarity: the character counts of the original texts are incremented by one and the score is based on the ratio of longer to shorter
- In case of dictionary absence: some steps are taken as follows:
    - The rough translation will be the source text itself
    - Sentence-level similarity will be surface identity of words
    - Simple dictionary is bootstrapped on this initial alignment

  - Collects one-to-one alignment with score above fixed threshold

- Moore's algorithm:
  - Initial alignment is constructed based only on sentence length similarity
  - IBM model 1 is trained on likely matched sentence pairs

So finally the similarity is measured using this translation model + sentence length similarity

4.5. CTK

- Champollion was initially developed for aligning Chinese-English parallel text; it was later ported to other language pairs.
- Supports Arabic-English
- Difference between Champollion and Other tools:
  - First, it assumes a noisy input, that a large percentage of alignments will not be one to one alignment, and that the number of deletions and insertions will be significant
  - Second, Champollion differs from other lexicon-based approaches in assigning weights to translated words

- What is the benefit of using lexicon in the sentence aligners?
  - First, translated words are identified by using entries from a translation lexicon
  - Second, statistics of translated words are then used to identify sentence correspondences

- Tokenizers:
  - CTK tokenizes both sides of the parallel text before computer the optimal alignment
  - In morphological complicated languages, such as English-Arabic, it splits sentences into words by white space then applies a light stemmer
  - Our experiment did not use stemmer

5. PYTESTER

It is Python test framework for text alignment tools, the goal of pyTester is to automate the process of testing different text alignment tools with the same unit tests, Its modular structure allows any researcher to add his own text alignment tool to the test framework easily.

PyTester provides reporting capability, which helps in doing the performance comparison between text alignment tools, it also allows adding more test cases easily.

It accepts XML file for the test cases, here is the xml format:

```
<?xml version="1.0" ?>
<data>
  <test id="0">
    <description>
      1000 random phrase from UN corpus without seperators
    </description>
    <source>
      arabic.small.10.txt
    </source>
    <target>
      english.small.10.txt
    </target>
    <golden>
      arabic_english_randomSelected_UN_6399_sentence.tmx
    </golden>
  </test>
…
</data>
```

PyTester structure consists of three layers, one layer for preparing the text for the proper input format for each tool, second layer for executing the tool, and third layer for analyzing the aligned text and evaluating it against the parallel corpora used.

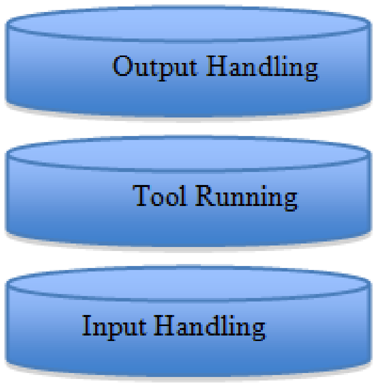

Figure 6: PyTester structure

If you want to add any text alignment tool to our test framework, you have to implement one or more of PyTester's layers according to your tool behavior. You can easily configure PyTester to your needs, you can add your own test cases, adjust the waiting time for each tool to execute or may be even add more criteria in your performance comparison test. Pytester is opensource project, you can get the code from https://github.com/johnnabil/pyTester

Experiment Conditions

I run PyTester with the following text alignment tools:

1) Hunalign
2) GMA
3) Vanilla
4) Bilingual Sentence Aligner
5) CTK (Champollion)

Before introducing my testing strategy, let me first define the meaning of unit test in our context, Unit test is pair of text files, one represents the source language (i.e. Arabic) and other one represents target language (i.e English), I generate those files from parallel corpora, so I have the golden reference for my test (i.e. the file that contains the right alignments).

Our Experiment is done using unit cases generated from the following parallel corpora:

1) UN parallel corpus
2) Meedan parallel corpus
3) Mozilla parallel corpus

Out testing strategy is divided into three phases:

1. Generating unit tests by randomly selecting chunks from the parallel corpus and Analyzing the relation between number of text chunks and time elapsed in the alignment process, average aligned chunk length and number of failed alignments
2. The same as previous phase but we add spurious chunks that do not have corresponding alignments in the other language. We begin by putting 1% of the number of chunks as spurious till 20 % of the text contains spurious chunks.
3. We run the same previous configuration but on French-English unit tests.

A simple Python script generator was made for generating those unit tests from the corpora mentioned above and inserting spurious phrases and variable punctuations between the alignments, this generator is included in the pyTester source code.

6. RESULTS

I run PyTester with all text alignment tools, it generates a report for each tool after running, and it calculates some criteria for each test case, as following:

1. Passed alignments (successful aligned text pairs)
2. Failed alignments (bad aligned text)
3. Time taken (in seconds) for each test case
4. Average English chunk length
5. Average Arabic chunk length
6. Average error rate
7. Whether the whole test case is completed successfully without any errors.

Here is an overview comparison between the tools after running our unit tests

Table 1: Comparison between the five alignment tools in English-Arabic sentences alignment

	Average Arabic Aligned Chunks' Length (In words)	Average English Aligned Chunks' Length (In words)	Average Time Taken (in seconds)	Average Error Rate
Bilingual Sentence Aligner	32.15	36.13	57	51%
GMA	33.16	36.99	19.40	74%
Hunalign	31.20	35.04	26.08	65.5%
Vanilla	31.5	35.65	27.38	73%
CTK	30	34	48	67.6%

As we can see Bilingual Sentence Aligner has the best accuracy between tools, but the longest execution time. In Vanilla we need special preprocessing on the text before running Vanilla.

Here are some graphs that describe the relation between the size of unit tests (number of text chunks) and time taken and number of failed alignments

1) Phase 1 of testing (without spurious chunks)

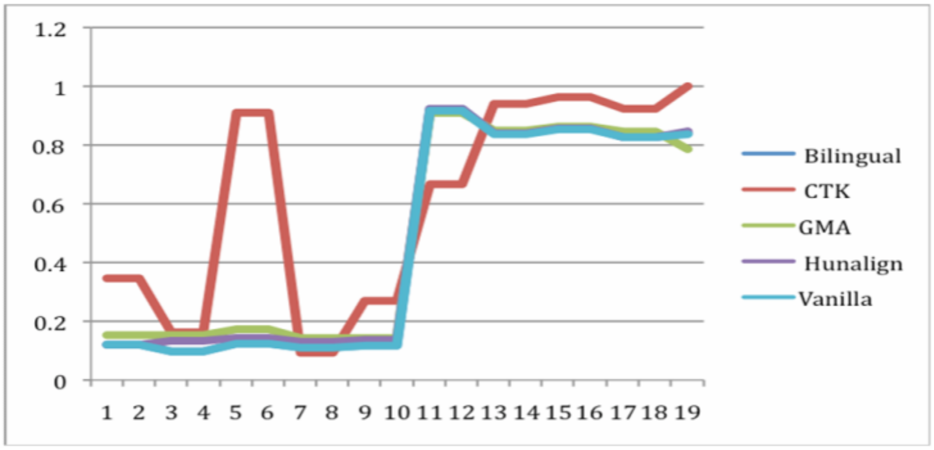

Figure 7: Error percentage in aligning Arabic-English pair

As you can see, the relation between the number of chunks and time taken in Vanilla and Hunalign is near to be constant, where it is linear in the case of CTK.

2) Phase 2 of testing (with spurious chunks).

Table 2: Comparison between the average error rate in each alignment tool

	Average Error Rate
Hunalign	70%
CTK	50%
Vanilla	95.5 %
GMA	94.9%
Bilingual	24 %

From the last table, we conclude that Bilingual sentence aligner is the most tolerant tool against the spurious chunks and Vanilla is least tolerant tool against the spurious chunks.

3) Phase 3 of testing (with French-English unit tests).

Table 3: Comparison between the five alignment tools in English-French sentences alignment

	Average French Aligned Chunks' Length (In words)	Average English Aligned Chunks' Length (In words)	Average Time Taken (in seconds)	Average Error Rate
Bilingual Sentence Aligner	47	43.5	66	46.6%
GMA	45.3	41.3	17.2	50%
Hunalign	47.8	43.5	21.2	46.6%
Vanilla	47.5	44.1	22.1	47%
CTK	41	37	48	51%

As we realize, when we use the same tools in testing French-English unit tests, we have better accuracy in all of them.

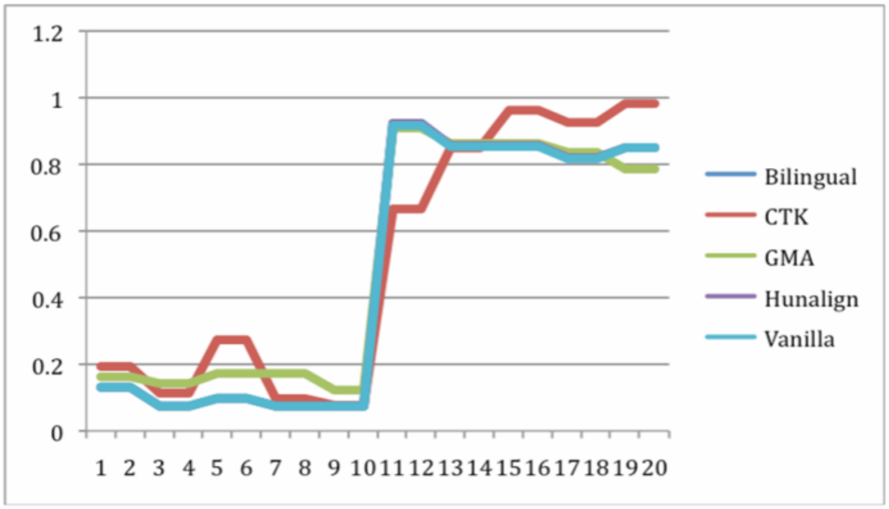

Figure 8: Error percentage in aligning French-English pair

7. CONCULSION

After running all of the most popular tools used in text alignment, I ended with some notes about them

- The most accurate tool is Hunalign and does not need any preprocessing
- The shortest execution time is done by GMA but it needs some special preprocessing however the time of this preprocessing is negligible.
- Regarding the error tolerance, CTK is the best alignment tool among the other 4 tools.

- We can select the suitable tool according to our nature of data. If our data contains a lot of spurious data we have to use tool like CTK to get good accuracy.
- If we cannot decide the how many spurious chunks are in our text, we can choose like Bilingual sentence aligner, which has average performance in the failed alignments and time taken. And also average tolerance against errors.

The overall accuracy for the tools in aligning French-English pairs is better than aligning Arabic-English pairs which gives us the opportunity to enhance this accuracy for Arabic in the future.

REFERENCES

[1] Ahrenberg, L., Andersson, M., Merkel, M. , 1998 , A simple hybrid aligner for generating lexical correspondences in parallel texts. Proceedings of the 17th international conference

[2] Danielsson, P., Sprakbanken, D., 1997, Practical presentation of a vanilla aligner, STAR '01 Proceedings of the ACL 2001 Workshop on Sharing Tools and Resources , Volume 15

[3] Fung, P. , Church, K. W. 1994, K-vec: a new approach for aligning parallel texts. Pro- ceedings of the 15th conference on Computational ,

[4] Gale, W. A. , Church, K. W. 1993, A program for aligning sentences in bilingual corpora, Computational Linguistics

[5] Ma, X., Champollion, 2006, A robust parallel text sentence aligner, LREC 2006: Fifth International Conference on Language Resources and Evaluation, Genova, Italy

[6] Melamed, D., 1999, Bitext maps and alignment via pattern recognition, Journal Computational Linguistics Journal, Volume 25.

[7] Moore, R., 2002, Fast and accurate sentence alignment of bilingual corpora, AMTA '02 Proceedings of the 5th Conference of the Association for Machine Translation in the Americas on Machine Translation.

[8] Salameh, M., Zantout, R., Mansour, N., 2011, Improving the accuracy of English-Arabic statistical sentence alignment, The International Arab Journal of Information Technology.

[9] Shannon C. E., 1948 ,A mathematical theory of communication, Bell System Technical Journal, vol. 27, pp. 379-423 and 623-656

[10] Tiedemann, J., Combinig ,2003, Clues for word alignment, EACL '03 Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics, Volume 1

[11] Varga, D., Halacsy, P., Kornai, A., Nagy, V., Nemeth, L., Tron, V., 2005, Parallel corpora for medium density languages, Natural Language Processing (RANLP),

Authors

John Nabil, is post-graduate student at the Arab Academy for Science, Technology & Maritime Transport, Cairo, Egypt, graduated from faculty of computer science and information system, Ain shams university, Cairo, Egypt in 2007, He has interests in NLP, machine learning and Image processing.

Dr. Mohamed Waleed Fakhr, Ph.D. 1993, University of Waterloo, in minimum complexity neural networks. Worked at Nortel, Montreal, Canada, in the speech research lab, 1994-1999, Professor in the Arab academy for science and technology, since 1999Currently, Professor at the university of Bahrain. Doing research in the areas of language processing, speech recognition, machine learning and compressed sensing.