TMSERVER
A Translation Memory Add-On for the Parsit Server

System Guide
**Introduction**

A Translation Memory (TM) is a database of text segments in a source language and their translations in one or more target languages. These text segments could be paragraphs, sentences or phrases.

Since the use of phrasal text segments would lead us into the domain of Phrase-Based SMT, the present system mainly focuses on sentences as the text segments. For certain domains where the same or very similar sentences occur very frequently, this approach can boost the accuracy of any Machine Translation system.

For example a rule-based system that translates SMS messages can benefit from a TM attached to it.

To this end, the system first tries to translate the user’s request by performing a fuzzy search on the TM database and translates if a near-exact match is found. On search failure the system delegates the request to the rule-based system. Thus the system adds an additional layer between the client and the rule-based server system.

---

![Diagram](image.png)

*Figure 1 Top Level System Architecture*
System Design

The system has the following main components:

![Diagram of System Design]

Figure 2 System Design of TM Server

The components are described below:

**TM Database**

There are many standards for storing TMs. The system stores the TM in the XML:TM standard format. The format was chosen for its flexibility, reliability and portability.
**Similarity Engine**

The similarity engine is assigned the task of finding similarity between words and sentences.

Word similarity is based on information content of character trigrams in the words being compared, as proposed by Lin. The information contents are learned from a large corpus. Various approaches were tried including String Equality, Stem Equality, Levenshtein distance or Edit Distance, Wu-Palmer's Measure, Lin's Semantic Measure besides the chosen approach. The approach was chosen for its high accuracy, high speed and portability.

Formally the word similarity measure between word \( w_1 \) and \( w_2 \) may be written as

\[
\text{sim}_{\text{word}}(w_1, w_2) = \frac{2 \times \sum_{t \in \text{tri}(w_1) \cap \text{tri}(w_2)} \log(p(t))}{\sum_{t \in \text{tri}(w_1)} \log(p(t)) + \sum_{t \in \text{tri}(w_2)} \log(p(t))} \tag{1}
\]

where \( \text{tri}(x) \) is the set of trigrams in \( x \). and \( p(t) \) is the probability of a trigram as learned from the corpus.

The \( p(t) \) values are smoothed for unknown character trigrams. This measure may be calculated in time linear to the length of the words being compared.

Words that are recognized as patterns of numbers, names etc. return a score of 1 if both the words have the same pattern.

Formally the word similarity measure using pattern information between word \( w_1 \) and \( w_2 \) may be written as

\[
\text{sim}_{\text{pat}}(w_1, w_2) = \begin{cases} 
1 & \text{if } \text{pat}(w_1) = \text{pat}(w_2) \\
\text{sim}_{\text{word}}(w_1, w_2) & \text{else}
\end{cases} \tag{2}
\]

where \( \text{pat}(x) \) is a function that returns the pattern for a word if any.
Once we are able to quantitatively compare words, we can align a pair of sentences to each other in order to maximize the alignment probability. Thus sentence similarity may be seen as an optimization problem or more specifically as an instance of the classical assignment problem, where the assignment between the words of two given sentences is maximized. Also we must weight sequences of matching words to be more important than single word matches.

These considerations lead to the following definition of the sentence similarity function between two sentences \( s_1 \) and \( s_2 \).

\[
\text{sim}_{\text{sen}}(s_1, s_2) = \frac{\text{align}_{\text{sen}}(s_1, s_2)}{\text{max} \left( |s_1|, |s_2| \right)}
\]

\[
\text{align}_{\text{sen}}(s_1, s_2) = \arg \max \sum_{w_i \in s_1, w_j \in s_2} \text{sim}_{\text{pat}}(w_i, w_j | w_{i-1}, w_{i+1}, w_{j-1}, w_{j+1})
\]

The Hungarian Algorithm may be employed to perform the above maximization in cubic time.

**Linguistic Processor**

The linguistic processor performs the following NLP tasks:

- Sentence Detection and Tokenization
- Textual parsing
- Linguistic parsing
- Pattern recognition
- Pattern Alignment

**Import**

The import utility accepts parallel sentences as raw text, processes them using the linguistic processor and finally saves them in the standard XML:TM format.
**Cluster**

The cluster utility uses the similarity engine to cluster sentences based on an optimal threshold found experimentally. This increases the speed of the search application and reduces memory requirements.

**Search**

This component is assigned the task of quickly searching the TM database and returning the top N translations. It uses a map of sentence clusters created automatically by using the Cluster utility. Each sentence cluster is mapped to each noun or verb in it.

**Evaluation**

The paraphrasing corpus was constructed to evaluate sentence similarity. It contains a set of manually made sentence clusters. The sentences in each cluster say the same thing in different ways.

The following figure shows a snapshot of the corpus.

*Either is highly inflammable.*
*Any of the two are highly inflammable.*

*This last act of tyranny inflamed the Russian populace.*
*This last act of domination through threat of punishment and violence had caused Russian people to get infla*
*This final act of tyranny heated the Russian community.*

*He inflamed.*
*His blood was inflamed.*
*He got angry.*

Figure 3 A snapshot of the Paraphrase Corpus

Currently there are about 3200 clusters containing more than 11000 sentences. The size of the corpus is large enough for evaluation purpose, although to perform machine learning of any kind, a much larger corpus would be needed.

Various similarity measures were compared to each other at varying thresholds and the F-Measure was calculate from the precision and recall. The resulting chart is shown below.
Future Work

A system to automatically extract a paraphrasing corpus from news headlines using a popular search engine is underway. With a larger corpus, it will be possible to perform machine learning using the well-known Statistical MT techniques by applying the constraint that a given word may only be aligned to its string equal word if such a candidate exists in the sentence pair.

The web may be used in other intuitive ways to learn in search for a better similarity measure.