

# **Introduction to Translation Memory and Machine Translation**

**Dr. Mark Ritter**  
**Chief Editor**  
**McElroy Translation Company**

---

**Introduction to Translation Memory  
and Machine Translation**

**Dr. Mark Ritter**  
**Chief Editor**  
**McElroy Translation Company**

# Course Objectives

- Understand the structure of a translation memory system
- Be able to set up and save a translation memory
- Apply a translation memory to a *Word* document
- Explore use of translation memory for HTML and more complex file formats

# Overview

- I. Principles of translation memory
- II. Demonstration on a *Word*® document  
(*Translator's Workbench*)
- III. Demonstration on an HTML file  
(*TagEditor*)
- IV. Translation memory pro and con

# Translation Memory vs. Machine Translation

- MT replaces human translators
- TM leverages the skill of human translators
- TM can work with virtually any language or subject; MT cannot
- MT is not ready for prime time
- TM = computer-assisted translation

# I Principles of Translation Memory Systems

- Translation memory vs. machine translation
- Structure of TM systems
  - Filtering of source
  - Segmentation
  - Editing / translation with memory
  - Updating memory
  - Outputting of final formatted translation

# TM Stage 1: Filtering

What needs to be translated here?

The TM preprocessor removes and saves formatting codes and then presents the **translation segments** to the translator.

## Raw file

```
<String `Table of Contents`>  
> # end of ParaLine  
> # end of Para  
<Para  
<Unique 948424>  
<Pgftag `ChapterTOC`>  
<PgfnString `1\t`>  
<ParaLine  
<TextRectID 21>  
<Marker  
<MType 8>  
<MTypeName `Hypertext`>  
<MText `openObjectId chapter1.fm:2 261488`>  
<MCurrPage `i`>  
<Unique 948423>  
> # end of Marker  
<String `Road Safety Education 1`>  
> # end of ParaLine
```

## Filtered output

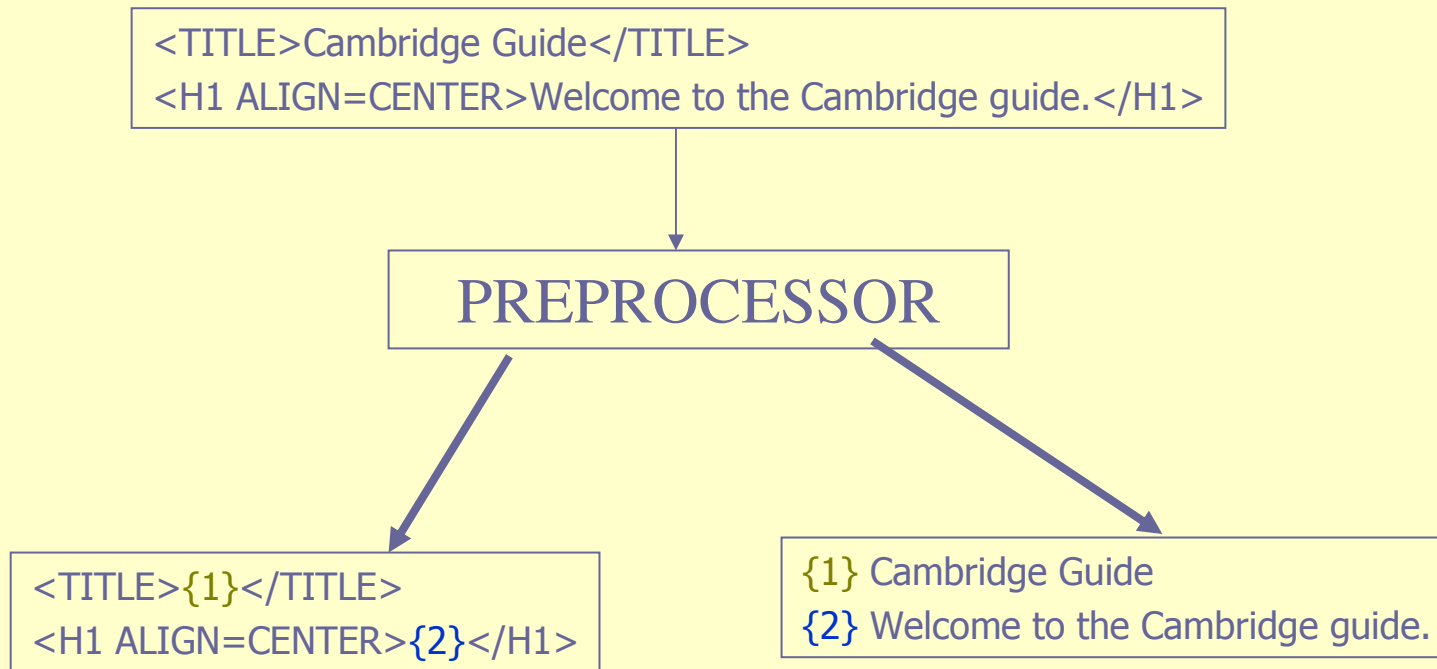
```
...  
[TS 1|Table of Contents]  
...  
[TS 2 | Road Safety Education]  
...
```

# TM Stage 2: Segmentation

- probability of recurrence of linguistic units is inversely proportional to length
- an intermediate length is likely to yield the largest number of useful matches
- for an efficient algorithm, segment boundaries must be automatically detectable, so

**TM SYSTEMS BREAK SEGMENTS AT SENTENCE BOUNDARIES OR END-OF-LINE CHARACTERS [stop characters]**

# The Filtering Process



# Filtering, II

- Some TM systems perform this process for all translation units in the source file (or even in several files at a time) before the user begins translating.
  - The list of translation units (= the right box in the preceding diagram) is the work file which the translator processes with the aid of the TM.
  - Systems using this approach include *Transit*, *SDLX* and *DejaVu*

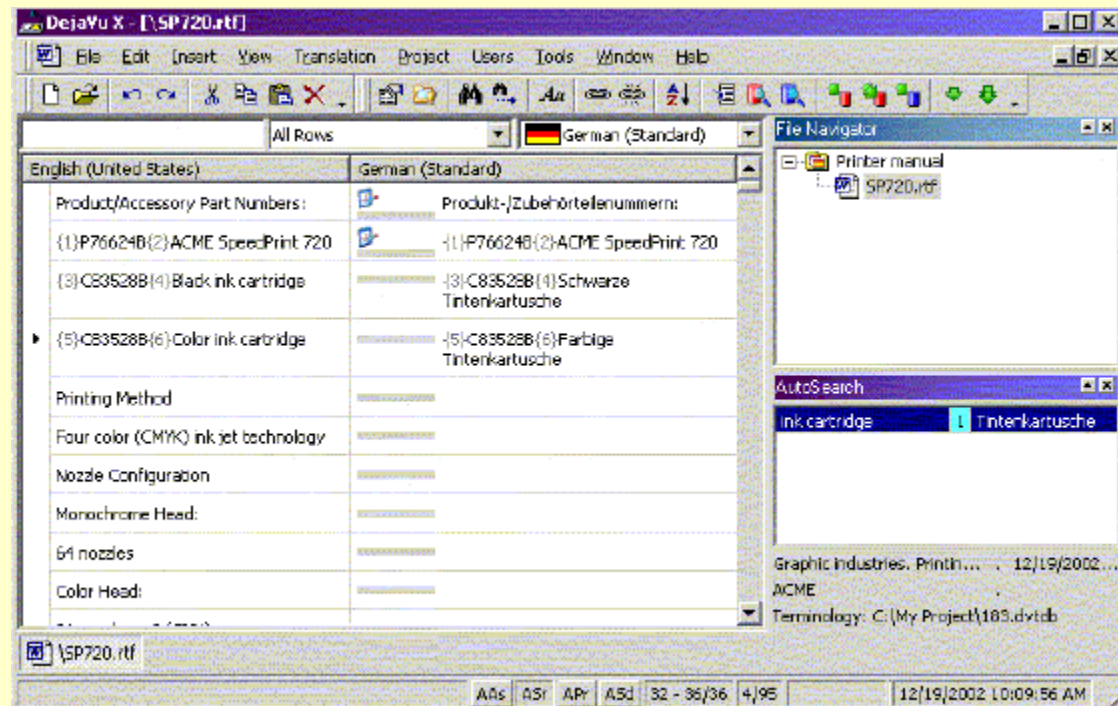
# Filtering, III

- The other approach, used by *Trados*<sup>TM</sup>, is more interactive.
  - The system moves through the file, popping one translation unit after another into an edit box.
  - This interface will be examined in detail in part II.

# Editing with the TM

- For both approaches, the editing work is basically the same for each TU:
  - Translate it, if the TM provides no suggestion
  - Accept the suggestion of the TM or
  - Modify the suggestion of the TM
- Whatever the choice it will now be available in the TM for subsequent translation units.

# Editing UI of a Batch-Mode TM



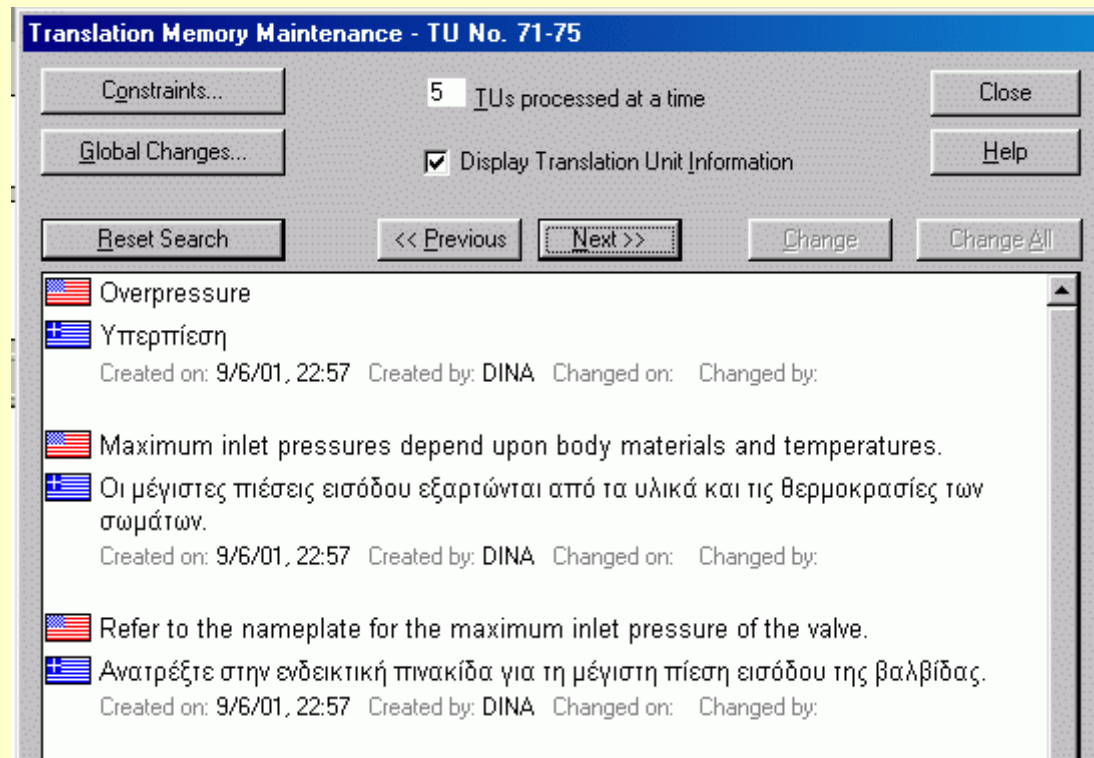
# Translation Memory Defined

- A translation memory is a searchable collection of *translation units*
- A translation unit consists of the following:

Source Segment | Target Segment | Housekeeping Data\*

\* The housekeeping data may include language identifiers, dates created/used/modified, user IDs, client IDs, subject matter codes and other features as configured by the user.

# Example of a Real TM (*Trados*)

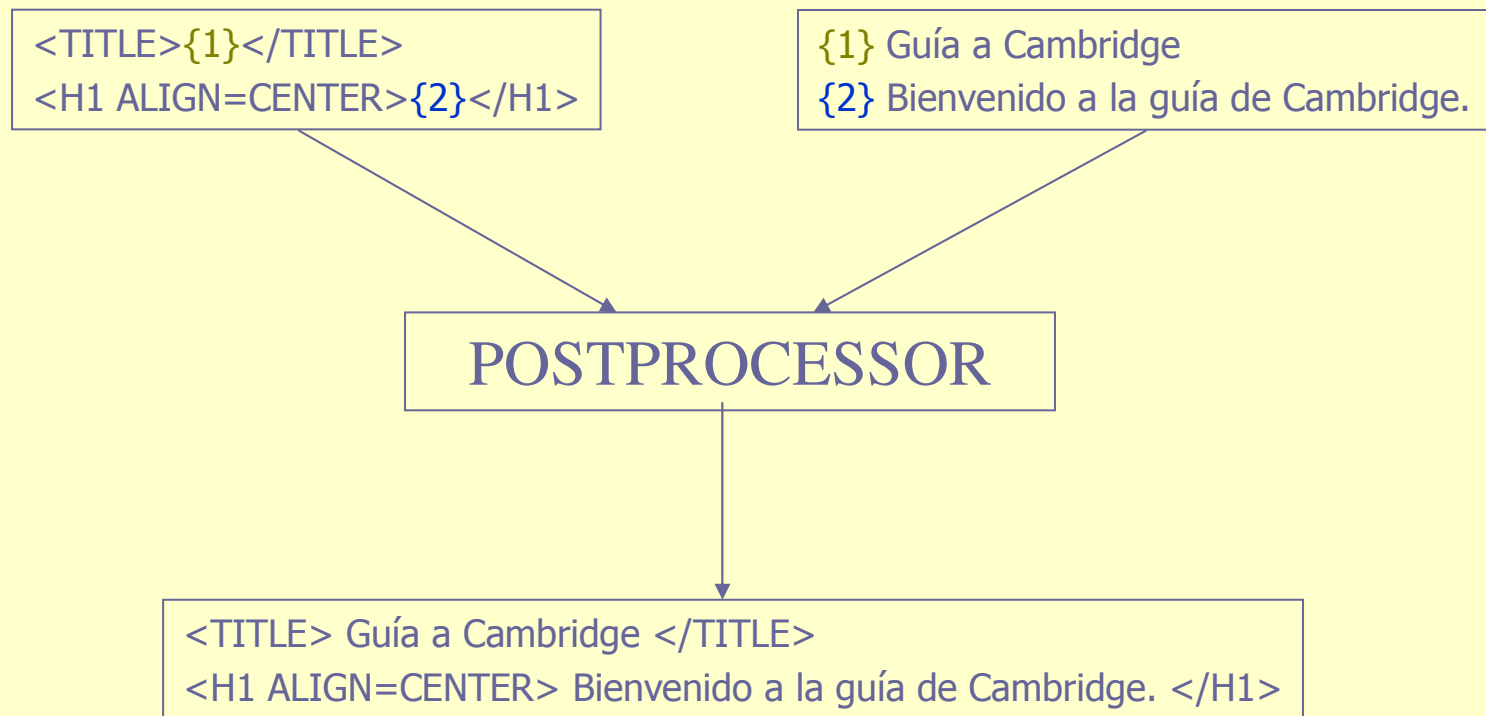


[as viewed from the **Maintenance** window of *Translator's Workbench*]

# Post-processing

- Reassembling the deliverable formatted file from the translated filtered file
  - Batch-mode TMs typically run a separate subprogram to transform the finished bilingual editing file into a target language file in the original format.

# Finishing Up



# Post-processing, II

- For Trados™, TM entries are partially hidden in the work file. They can (and should) be stripped out (by the translator, the vendor, or the client) and saved, leaving behind the finished document. More details on the Trados™ method in part II.

# Review of TM Process

- PREPROCESSING [automatic]
- TRANSLATION ALGORITHM [human]
- POSTPROCESSING [automatic]

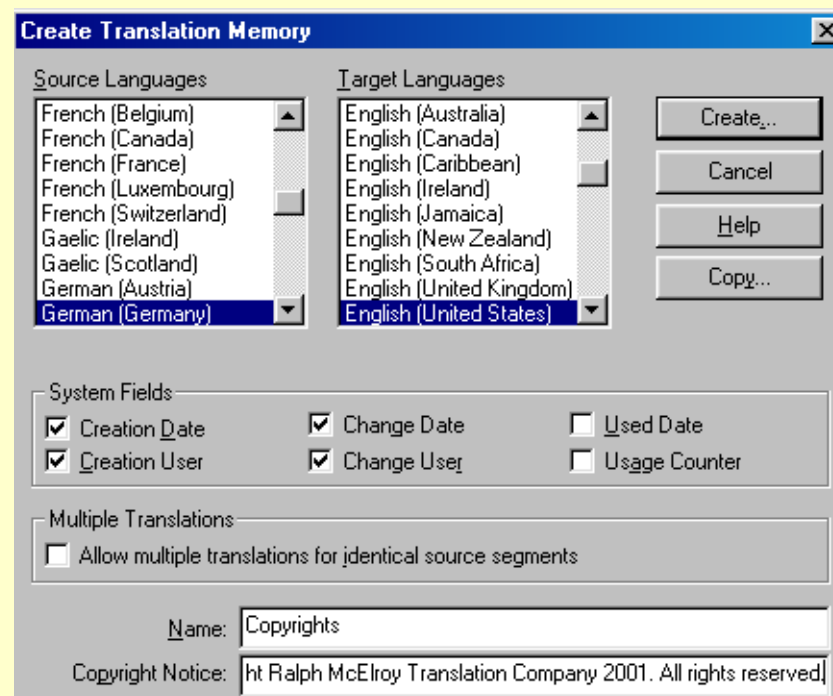
# Translation Memory Packages

- Trados (Trados Corporation)
- DejaVU / DVX (Atril)
- Transit (Star International)
- SDLX (SDL International)
- Above are all for the PC—THERE ARE NO ENTERPRISE-LEVEL SUITES FOR MAC OS, LINUX OR UNIX

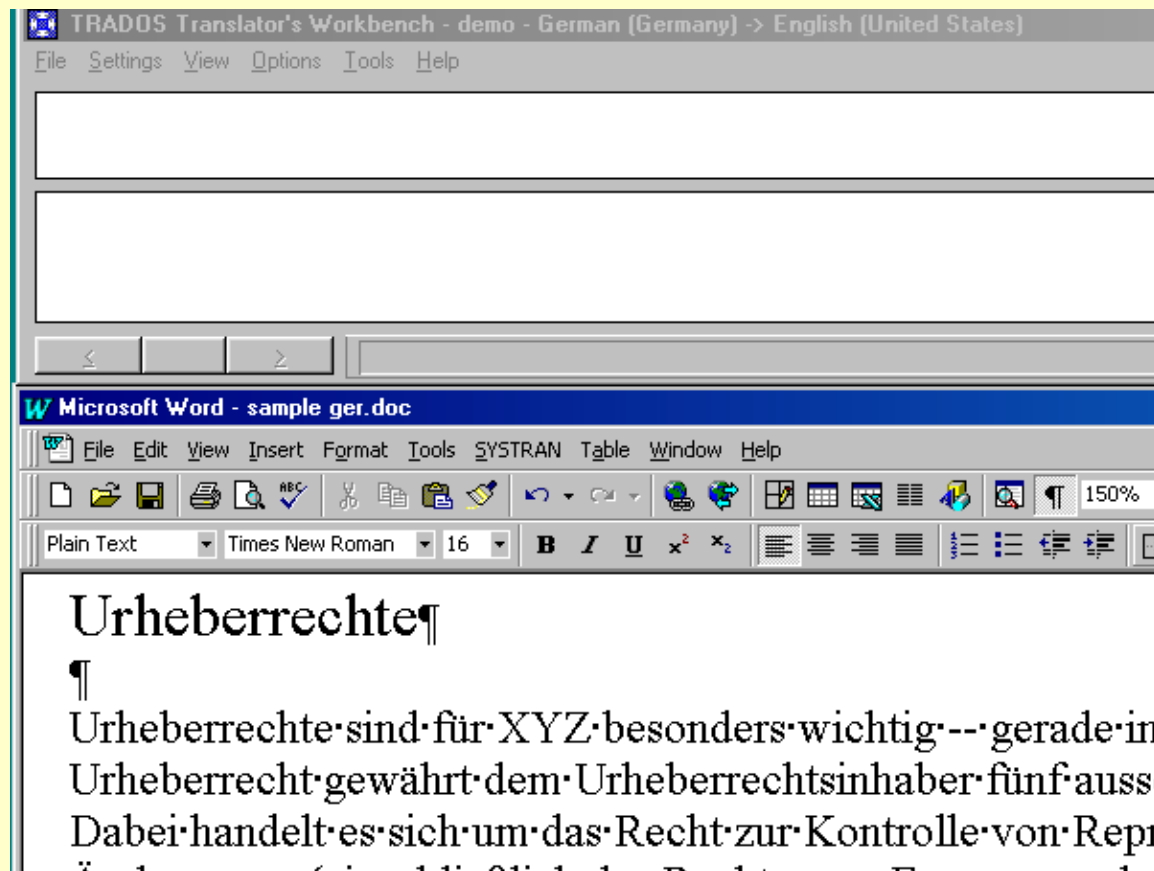
## II Application of Trados to a Word Document

- Memory setup
- Translation of the document unit-by-unit
  - No match: translate conventionally
  - 100% match: adopt
  - Fuzzy match: adopt with edits
- Post-processing

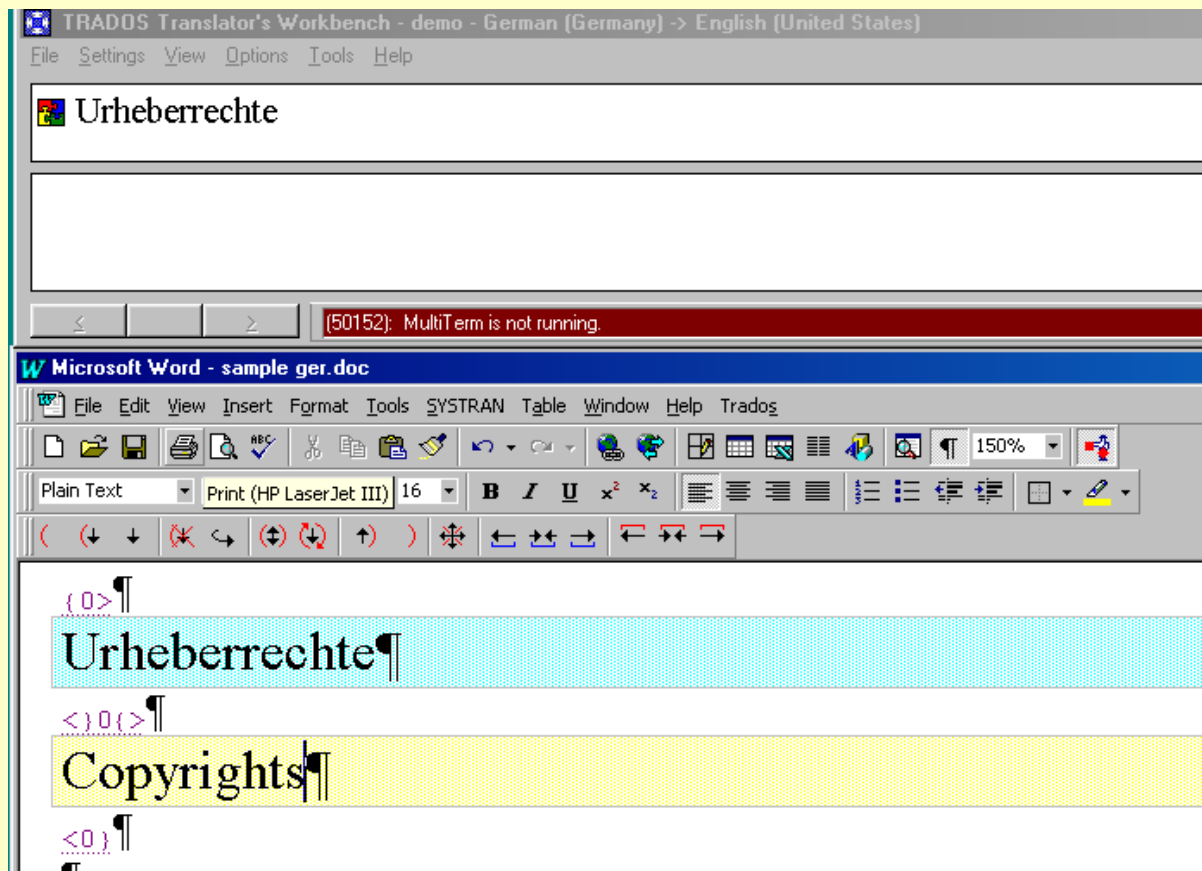
# Memory setup



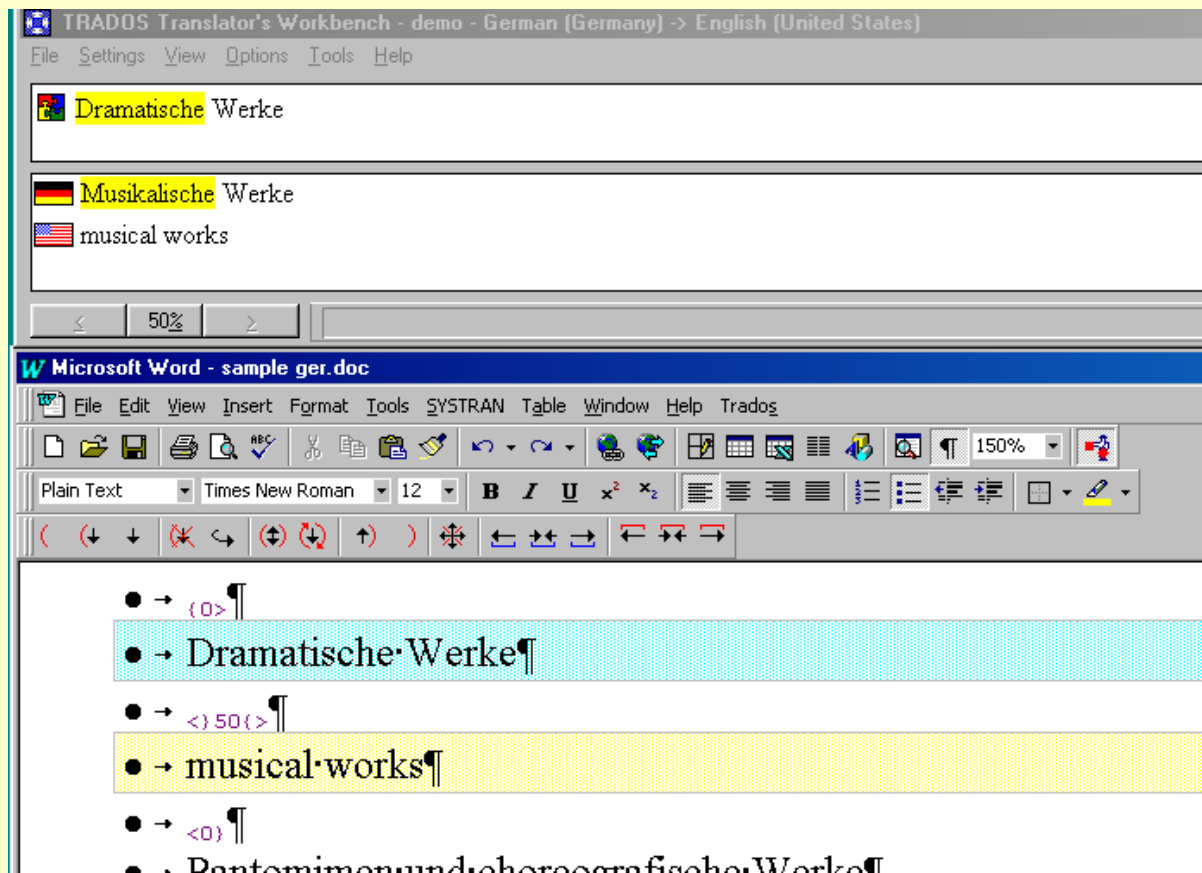
# Memory and Word launched



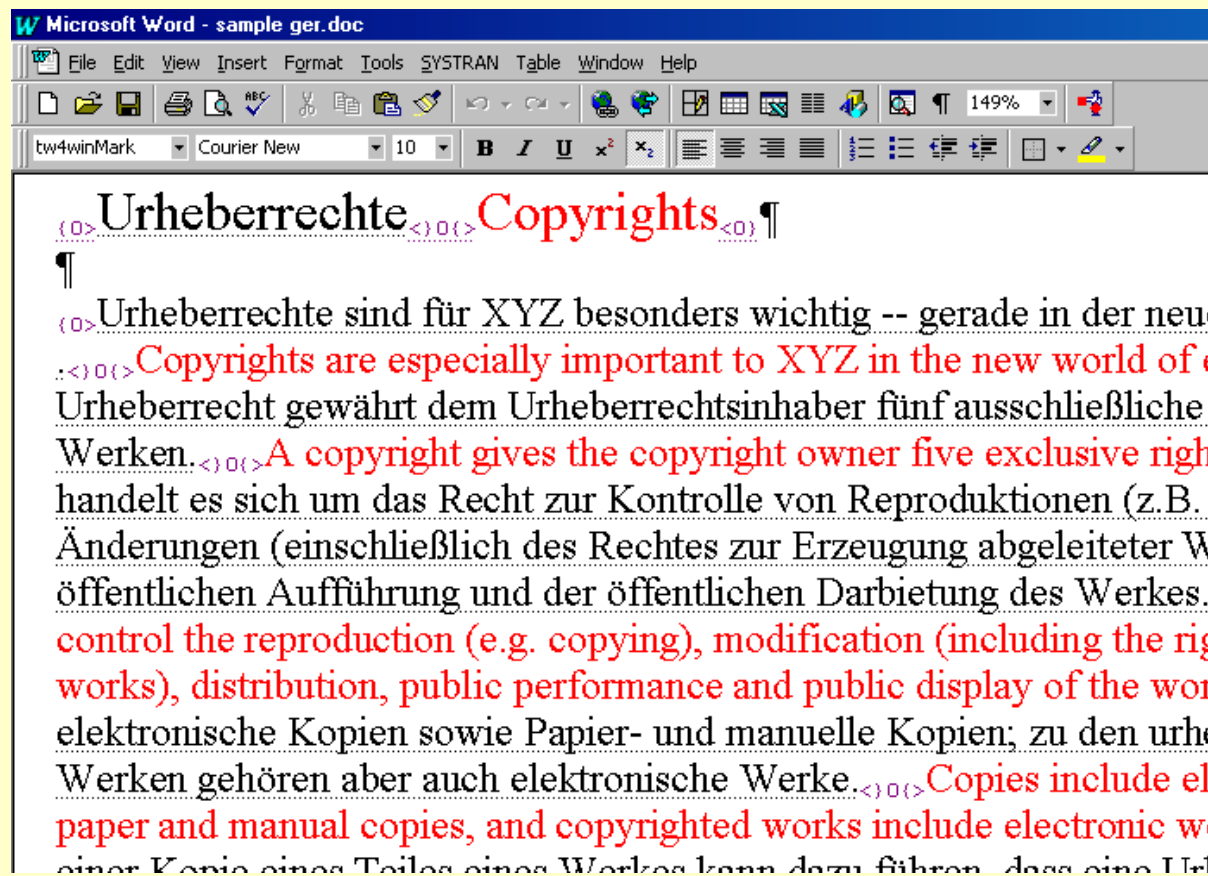
# The Edit window



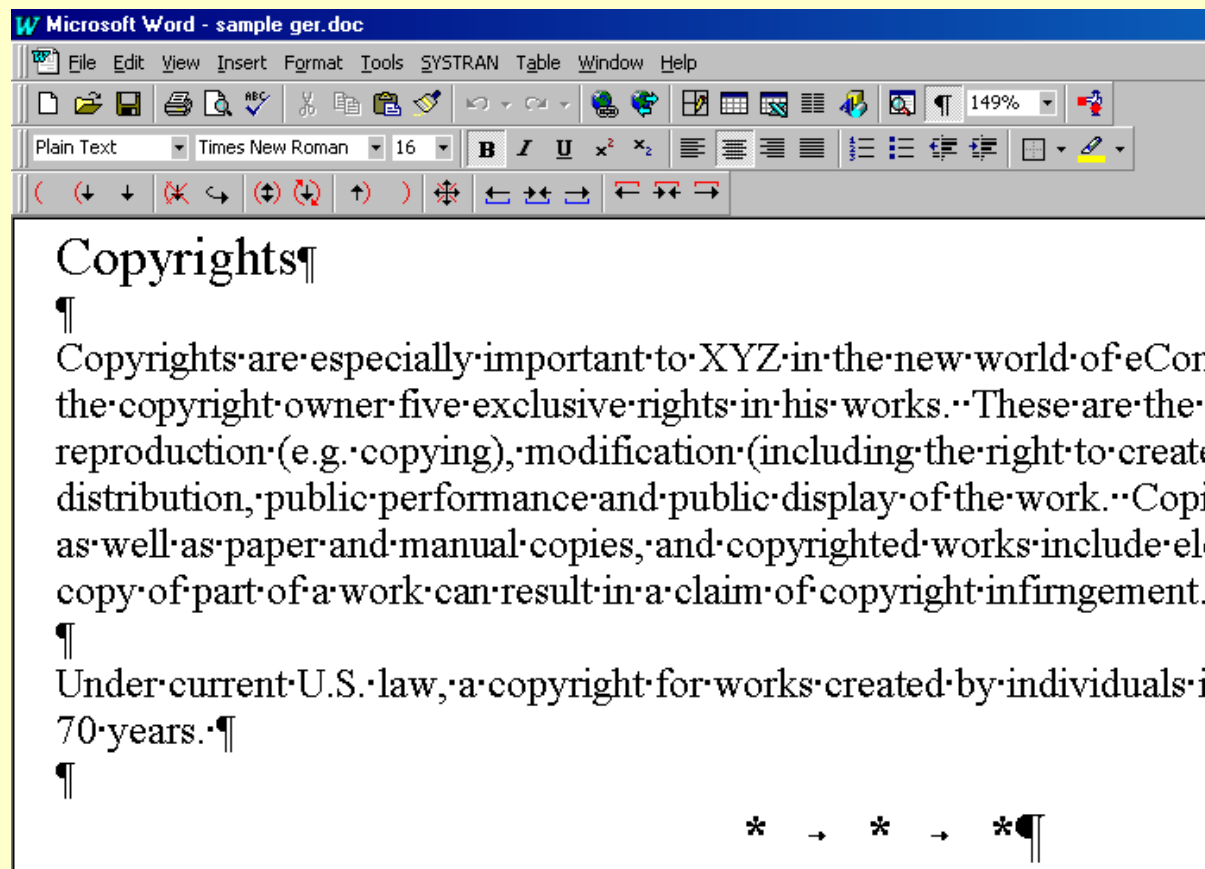
# Example of a fuzzy match



# The final bilingual file



# The cleaned translation



# III Application of Trados to a Web page

- The problem with “tagged” texts
- Tag Editor--the Trados filtering tool
- Tag Editor works similarly to Word
- Protecting tags
- Real-time previewing
- Post-processing as usual

# What needs translation here?

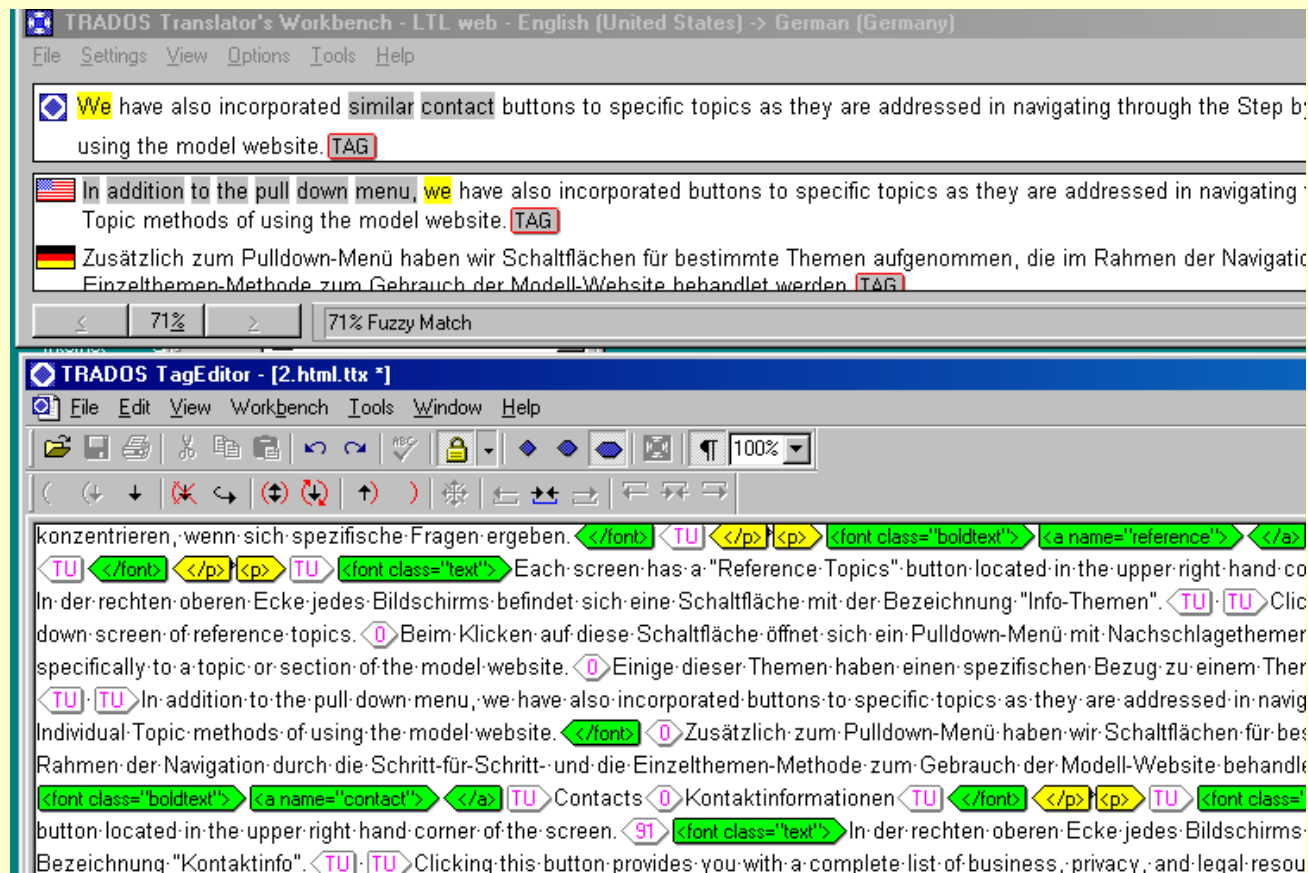
```
border="0" align="left"></td>
28                                     <td align="right">
29                                         <table cellpa
cellspacing="0" border="0">
30                                             <tr>
31
href="contact.html"></a></td>
32
valign="bottom" align="right">
33
name="form0" action="#" method="post">
34
name="nav" size="1"
onchange="window.location=document.form0.nav.options[document.form0.nav.selec
35
<option value="">Reference Topics
36
<option value="ref_advertising.html">Advertising
37
<option value="ref_biz_continuity.html">Business Continuity
38
<option value="ref_collecting_info.html">Collecting Information
39
<option value="ref_content.html">Content
40
<option value="ref_copyright.html">Copyright
41
```

# The Solution: Filtering

- For batch-mode TMs, the translatable text is extracted and dumped into one column of a table for editing, as seen previously
- Trados<sup>TM</sup> uses a separate program, TagEditor, that runs together with the Translators Workbench memory manager. It offers a more graphical interface.



# Tag Editor and the TM



# Editing translatable text

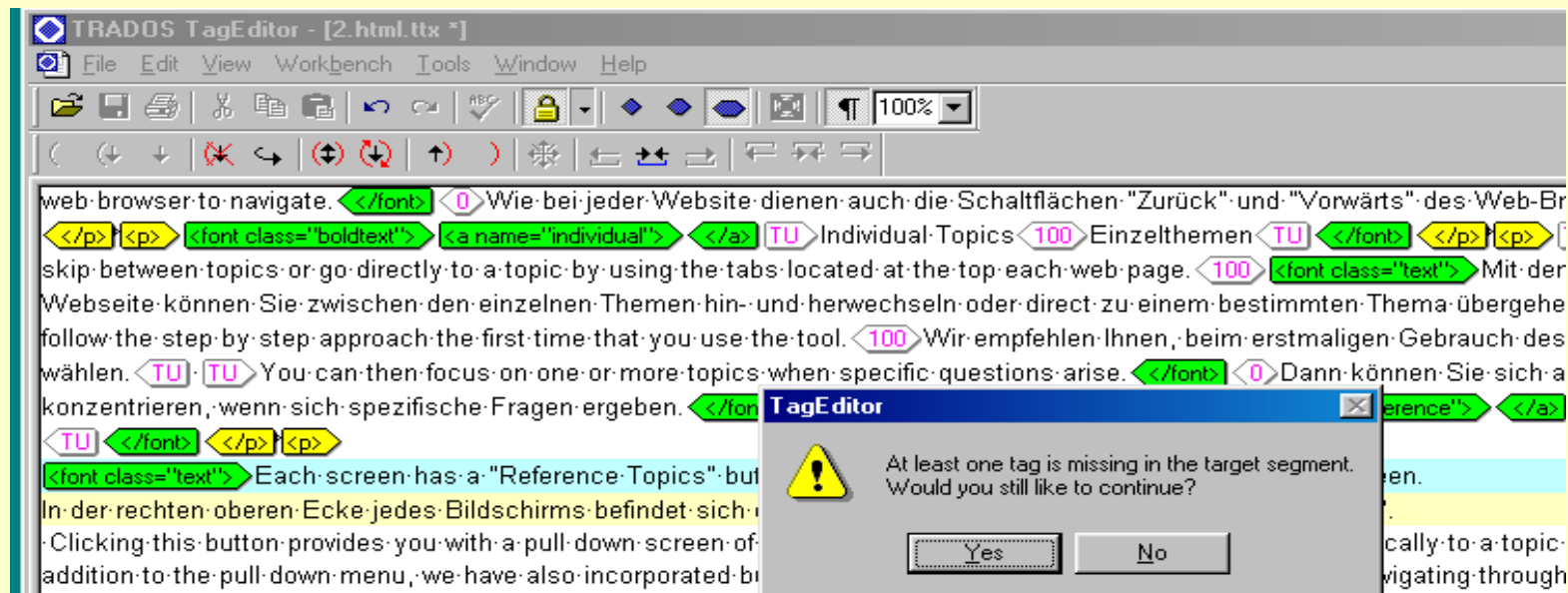
- Tag Editor pops up a window for each successive translation unit, including tags inside the unit
- The user is responsible for getting these copied, using the associated toolbar

We have also incorporated similar contact buttons to specific topics as they are addressed in navigating through the St  
the model website. </font>

Zusätzlich zum Pulldown-Menü haben wir Schaltflächen für bestimmte Themen aufgenommen, die im Rahmen der Navi  
Einzelthemen-Methode zum Gebrauch der Modell-Website behandelt werden. </font>

# Tag protection

- If the user fails to copy tags properly, Tag Editor issues a warning





# TagEditor pros and cons

- Because it's visual, real-time preview helps preserve look and feel of source
- The visibility of tags can alert the translator to problems with segmentation
- TagEditor can handle only one file at a time; batch mode can handle many small files at once.
- Not easy to update for new file formats

## IV Advantages of TM

- Translator need not have the source application
- Leveraging and integrated termbase increase consistency
- Existing memories can be reused for similar projects
- Revisions are much easier to handle

# Cautions

- Source must be electronic
- Can propagate errors if misused
- A memory is only as good as the maintenance it gets
- Not appropriate for all material
  - ultra-long sentences yield fewer matches
  - may tend to obscure nuances
  - output may become monotonous