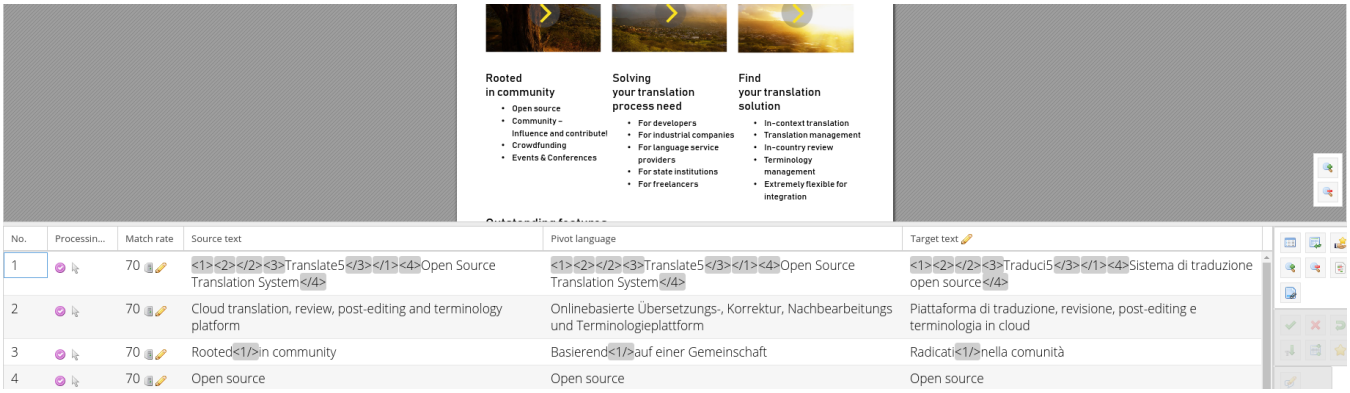
Using a relay / pivot / second source language

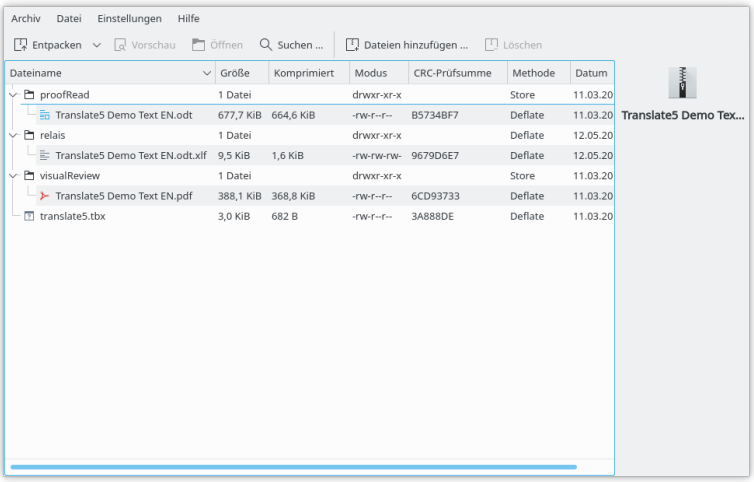
Translate5 supports the usage of a second source language, a so-called relay or pivot language:



Adding a relay language

Please see the attached import zip file as an example: pivot-test-es-de.zip (use English as the Relais/Pivot language, Spanish as the source language, and German as the target language)

1. Create a zip import file as usual
2. Provide a folder "relais" in the zip container
3. The relais folder has to contain the same file and folder structure as the workfiles folder, with the difference that the bilingual data has the desired relay language set as its target language
    a. The relais folder must always contain bilingual data, the workfiles folder may contain also non-bilingual data, which must then be imported with Okapi.
    In this case, a file "testfile.docx" in the work data in workfiles will match the bilingual file "testfile.docx.xlf" in the relais folder.



A short explanation of what is displayed in this screenshot of an import zip file:

> Additional info: proofRead is deprecated, use workfiles instead!

File	Purpose
workfiles/Translate5 Demo Text EN.odt	LibreOffice file to be translated, the source language is English. It is to be translated into the target language (Italian).
relais/Translate5 Demo Text EN.odt.xlf	A previous translation of the same file was made into German. The resulting bilingual file from that translation is now used as a relay language, making German the relay language in this case.

visualReview/Translate5 Demo Text EN.pdf	The Original LibreOffice file was converted to PDF, to be used as a VisualReview file.
translate5.tbx	Optional: A TBX file can be directly attached to a zip file. This is normally done by associating an already imported TermCollection as a language resource to the task.

Matching / Aligning of the relay and source file

To ensure that the content of the source file can be matched to the content of the relay file, the following algorithm is used:

1. As explained above, in order for the files to match the file names and folder structure must be identical in both the workfiles and the relais folder.
2. The relay file is parsed. For each relay segment, the segment in the work data is loaded
   a) either by the same ID, with the ID depending on the import file format.
   b) or by the same segment number in the task
3. In both cases, the relay source content of the segment must be identical to the source content of the segment to be translated.
   You can configure how the source content is to be compared, an explanation of which can be found below.

Matching examples

Translate5 imports the first file. After that, the relay file is parsed. Due to the same ID (in this case the ID=2) and the identical source text, translate5 is able to align the segments properly.

If no segment corresponding to ID=2 can be found, translate5 will try to load the segment of the same position is tried to be loaded. In this case, the source content must also be identical.

proofRead/Translate5 Demo Text EN.odt (converted to XLF by Okapi)

```
<trans-unit id="2" restype="x-text:h">
        <source xml:lang="en">Cloud translation, review, post-editing and terminology platform</source>
        <seg-source><mrk mid="0" mtype="seg">Cloud translation, review, post-editing and terminology platform<
/mrk></seg-source>
        <target xml:lang="it"></target>
</trans-unit>
```

Excerpt: relais/Translate5 Demo Text EN.odt.xlf

```
<trans-unit id="2" restype="x-text:h">
        <source xml:lang="en">Cloud translation, review, post-editing and terminology platform</source>
        <seg-source><mrk mid="0" mtype="seg">Cloud translation, review, post-editing and terminology platform<
/mrk></seg-source>
        <target xml:lang="de"><mrk mtype="seg" mid="0">Onlinebasierte Übersetzungs-, Korrektur,
Nachbearbeitungs und Terminologieplattform</mrk></target>
</trans-unit>
```

Used ID for segment loading depending on the file format

File Format	Used value as ID
XLF	trans-unit id + segmentnrInTask (the suffix is needed for uniqueness, since XLF may contain multiple files with the same trans-unit id)
SDLXLIFF	trans-unit id
Transit	trans-unit id
CSV	The values from the column configured as "mid" column
XlfZend (internal)	md5 hash of the long trans-unit id (a base64 encoded string)

Configuring how the source content is compared

In the configuration "runtimeOptions.import.relaisCompareMode" the flags "IGNORE_TAGS" and "NORMALIZE_ENTITIES" can be set.
Both are enabled by default.

IGNORE_TAGS: All tags are removed before comparison. To keep word boundaries, the tags are replaced with whitespace. Multiple whitespaces are merged into a single one.

NORMALIZE_ENTITIES: HTML Entities are decoded to enable comparison of " in one and " in the other source content.