Using multilingual content on the web to build fast finite-state direct translation systems

Mikel L. Forcada
Departament de Llenguatges i Sistemes Informàtics
Universitat d’Alacant
E-03071 Alacant, Spain

*Partially funded by CICyT (project TIC2000-1599-C02-02).
Index

The objective

The web as a source of translation units

Using subsentential translation units

Finite-state storage of TU databases

Updating the finite-state translation memory

A web-based application

Summary and outlook
Index

The objective

The web as a source of translation units

Using subsentential translation units

Finite-state storage of TU databases

Updating the finite-state translation memory

A web-based application

Summary and outlook
Index

The objective

The web as a source of translation units

Using subsentential translation units

Finite-state storage of TU databases

Updating the finite-state translation memory

A web-based application

Summary and outlook
Index

The objective

The web as a source of translation units

Using subsentential translation units

Finite-state storage of TU databases

Updating the finite-state translation memory

A web-based application

Summary and outlook
Index

The objective

The web as a source of translation units

Using subsentential translation units

Finite-state storage of TU databases

Updating the finite-state translation memory

A web-based application

Summary and outlook
Index

The objective

The web as a source of translation units

Using subsentential translation units

Finite-state storage of TU databases

Updating the finite-state translation memory

A web-based application

Summary and outlook
Index

The objective

The web as a source of translation units

Using subsentential translation units

Finite-state storage of TU databases

Updating the finite-state translation memory

A web-based application

Summary and outlook
Index

The objective

The web as a source of translation units

Using subsentential translation units

Finite-state storage of TU databases

Updating the finite-state translation memory

A web-based application

Summary and outlook
The objective
The objective

Use existing multilingual web content...
The objective

Use \textbf{existing} multilingual web content...

...to build and maintain a web-based, fast...
The objective

Use existing multilingual web content...

...to build and maintain a web-based, fast...

...direct machine translation engine...
The objective

Use existing multilingual web content...

...to build and maintain a web-based, fast...

...direct machine translation engine...

...for MT-less (or MT-poor) language pairs.
The web as a source of translation units
The web as a source of translation units

One can mine the web for bitexts.
The web as a source of translation units

One can mine the web for bitexts.

The web as a source of translation units

One can mine the web for bitexts.


Bitext identification involves bitext alignment.
The web as a source of translation units

One can mine the web for bitexts.


Bitext identification involves bitext alignment.

Alignment produces candidate translation units (TU).
The web as a source of translation units

One can mine the web for bitexts.


Bitext identification involves bitext alignment.

Alignment produces candidate translation units (TU).

Coverage may be large for certain language pairs, registers, and subjects.
The web as a source of translation units

One can mine the web for bitexts.


Bitext identification involves bitext alignment.

Alignment produces candidate translation units (TU).

Coverage may be large for certain language pairs, registers, and subjects.

Build a (web-based) translation-memory-like engine?
Using subsentential translation units/1

Customary translation memories use non-linguistic or weakly-linguistic alignment techniques:

- Sentence boundary determination,
- Format analysis,
- Proper-name identification, etc.

Result: segments are sentences (running text) or list items (menus, tables, etc.).

Coverage of sentence segments may not be large → approximate matching.
Using subsentential translation units/1

Customary translation memories use non-linguistic or weakly-linguistic alignment techniques:

- Sentence boundary determination,
- Format analysis,
- Proper-name identification, etc.

Result: segments are sentences (running text) or list items (menus, tables, etc.).

Coverage of sentence segments may not be large → approximate matching.
Using subsentential translation units/1

Customary translation memories use non-linguistic or weakly-linguistic alignment techniques:

- Sentence boundary determination,
Using subsentential translation units/1

Customary translation memories use non-linguistic or weakly-linguistic alignment techniques:

- Sentence boundary determination,
- format analysis,
Using subsentential translation units/1

Customary translation memories use non-linguistic or weakly-linguistic alignment techniques:

- Sentence boundary determination,
- format analysis,
- proper-name identification, etc.
Using subsentential translation units/1

Customary translation memories use non-linguistic or weakly-linguistic alignment techniques:

- Sentence boundary determination,
- format analysis,
- proper-name identification, etc.

Result:
Using subsentential translation units/1

Customary translation memories use non-linguistic or weakly-linguistic alignment techniques:

- Sentence boundary determination,
- format analysis,
- proper-name identification, etc.

Result: segments are sentences (running text)
Using subsentential translation units/1

Customary translation memories use non-linguistic or weakly-linguistic alignment techniques:

- Sentence boundary determination,
- format analysis,
- proper-name identification, etc.

Result: segments are sentences (running text) or list items (menus, tables, etc.).
Using subsentential translation units/1

Customary translation memories use non-linguistic or weakly-linguistic alignment techniques:

- Sentence boundary determination,
- format analysis,
- proper-name identification, etc.

Result: segments are sentences (running text) or list items (menus, tables, etc.).

Coverage of sentence segments may not be large → approximate matching.
Using subsentential translation units/2

Subsentential units may improve coverage. May be obtained from specialized subsentential aligning (Simard & Langlais 2001). . . or using statistical MT techniques (Marcu 2001). But may also introduce ambiguity:

• A careful balance is necessary: coverage vs. precision.

• Subject, register, and variety annotation crucial.
Using subsentential translation units/2

Subsentential units may improve coverage.
Using subsentential translation units/2

Subsentential units may improve coverage.

May be obtained from specialized subsentential aligning (Simard & Langlais 2001). . .
Using subsentential translation units/2

Subsentential units may improve coverage.

May be obtained from specialized subsentential aligning (Simard & Langlais 2001)...

...or using statistical MT techniques (Marcu 2001).
Using subsentential translation units/2

Subsentential units may improve coverage.

May be obtained from specialized subsentential aligning (Simard & Langlais 2001)...

...or using statistical MT techniques (Marcu 2001).

But may also introduce ambiguity:
Using subsentential translation units/2

Subsentential units may improve coverage.

May be obtained from specialized subsentential aligning (Simard & Langlais 2001). . .

. . .or using statistical MT techniques (Marcu 2001).

But may also introduce ambiguity:

- A careful balance is necessary: coverage vs. precision.
Using subsentential translation units/2

Subsentential units may improve coverage.

May be obtained from specialized subsentential aligning (Simard & Langlais 2001). . .

. . . or using statistical MT techniques (Marcu 2001).

But may also introduce ambiguity:

- A careful balance is necessary: coverage vs. precision.
- Subject, register, and variety annotation crucial.
Using subsentential translation units/3
Using subsentential translation units/3

Regular TMs use the same weakly-linguistic segmentation strategies when (pre)translating new text.
Using subsentential translation units/3

Regular TMs use the same weakly-linguistic segmentation strategies when (pre)translating new text.

But subsentential units may also be used to segment texts...
Using subsentential translation units/3

Regular TMs use the same weakly-linguistic segmentation strategies when (pre)translating new text.

But subsentential units may also be used to segment texts...

...both when translating new text...
Using subsentential translation units/3

Regular TMs use the same weakly-linguistic segmentation strategies when (pre)translating new text.

But subsentential units may also be used to segment texts...

...both when translating new text...

...and when aligning new bitexts.
Using subsentential translation units/4

If we have the TU ("web bitexts", "los bitextos en la red") and the bitext contains the sentence pair

"Web bitexts are a valuable resource.

Los bitextos en la red son un recurso muy valioso.

we obtain a candidate TU:

("

**are a valuable resource

son un recurso muy valioso"

)

A similar subsentential segmentation strategy must be designed for pretranslation.
An alignment example: If we have the TU

(‘‘web bitexts’’, ‘‘los bitextos en la red’’)
An alignment example: If we have the TU

(‘‘web bitexts’’, ‘‘los bitextos en la red’’)

and the bitext contains the sentence pair

Web bitexts are a valuable resource.
Los bitextos en la red son un recurso muy valioso.
An alignment example: If we have the TU

(‘‘web bitexts’’, ‘‘los bitextos en la red’’)

and the bitext contains the sentence pair

Web bitexts are a valuable resource.
Los bitextos en la red son un recurso muy valioso.

we obtain a candidate TU:

(‘‘are a valuable resource’’, ‘‘son un recurso muy valioso’’)

Using subsentential translation units/4
Using subsentential translation units/4

An alignment example: If we have the TU

(‘‘web bitexts’’, ‘‘los bitextos en la red’’)

and the bitext contains the sentence pair

Web bitexts are a valuable resource.
Los bitextos en la red son un recurso muy valioso.

we obtain a candidate TU:

(‘‘are a valuable resource’’, ‘‘son un recurso muy valioso’’)

A similar subsentential segmentation strategy must be designed for pretranslation.
Finite-state storage of TU databases/1

→ TU lookup very intensive!

A possible solution: finite-state transducers (FST):

TUs are input–output pairs: encoded as state paths in an FST.

Finite-state theory very well developed: e.g., minimization (common substring correspondences in TUs may share paths).
Finite-state storage of TU databases/1

TUs used to segment?
Finite-state storage of TU databases/1

TUs used to segment? → TU lookup very intensive!
Finite-state storage of TU databases/1

TUs used to segment? → TU lookup very intensive!

A possible solution: finite-state transducers (FST):
Finite-state storage of TU databases/1

TUs used to segment? → TU lookup very intensive!

A possible solution: finite-state transducers (FST):

TUs are input–output pairs: encoded as state paths in an FST.
Finite-state storage of TU databases/1

TUs used to segment? → TU lookup very intensive!

A possible solution: finite-state transducers (FST):

TUs are input–output pairs: encoded as state paths in an FST.

Finite-state theory very well developed: e.g., minimization
Finite-state storage of TU databases

TUs used to segment? → TU lookup very intensive!

A possible solution: finite-state transducers (FST):

TUs are input–output pairs: encoded as state paths in an FST.

Finite-state theory very well developed: e.g., minimization

(common substring correspondences in TUs may share paths).
Finite-state storage of TU databases/2

A typical way of using FSTs: Left-to-right, longest-match (LRLM: lex, chunkers, fixed-length-phrase parsers, etc.)

Example: we have two TUs: 
   ("a translation", "una traducción")
   ("a translation memory", "una memoria de traducción")

The text "Una memoria de traducción es un archivo..." will be matched by the second one.

If no match is found, skip one word and try again.
Finite-state storage of TU databases/2

A typical way of using FSTs:
Finite-state storage of TU databases/2

A typical way of using FSTs:

Left-to-right, longest-match (LRLM: lex, chunkers, fixed-length-phrase parsers, etc.)
Finite-state storage of TU databases/2

A typical way of using FSTs:

Left-to-right, longest-match (LRLM: lex, chunkers, fixed-length-phrase parsers, etc.)

Example: we have two TUs:

(‘‘a translation’’, ‘‘una traducción’’)
(‘‘a translation memory’’, ‘‘una memoria de traducción’’)

will be matched by the second one.

If no match is found, skip one word and try again.
Finite-state storage of TU databases/2

A typical way of using FSTs:

*Left-to-right, longest-match* (*LRLM*: `lex`, chunkers, fixed-length-phrase parsers, etc.)

Example: we have two TUs:

(‘‘*a translation’’,‘‘*una traducción’’)

(‘‘*a translation memory’’,‘‘*una memoria de traducción’’)

The text ‘‘*Una memoria de traducción es un archivo...’’’ will be matched by the second one.
Finite-state storage of TU databases/2

A typical way of using FSTs:

**Left-to-right, longest-match** (LRLM: lex, chunkers, fixed-length-phrase parsers, etc.)

Example: we have two TUs:

(‘‘a translation’’, ‘‘una traducción’’)

(‘‘a translation memory’’, ‘‘una memoria de traducción’’)

The text ‘‘Una memoria de traducción es un archivo...’’ will be matched by the second one.

If no match is found, skip one word and try again.
Finite-state storage of TU databases/3

("a translation", "una traducción")
("a translation memory", "una memoria de traducción")
Finite-state storage of TU databases/4

Letter transducers (LTs): the simplest FSTs.
Transitions: read one or zero characters, write one or zero characters.
May implement optimal input–output string alignment: "edit distance" (minimal insertions, deletions and substitutions).
Similar TUs likely to share similar alignments: path sharing.

LTs are isomorphic to finite-state automata:
• standard minimization (Hopcroft & Ullman 1979),
• lightning-fast implementations (Bentley & Sedgewick 1997).
Finite-state storage of TU databases/4

Letter transducers (LTs): the simplest FSTs.
Finite-state storage of TU databases/4

Letter transducers (LTs): the simplest FSTs.

Transitions: read one or zero characters, write one or zero characters.
Finite-state storage of TU databases/4

Letter transducers (LTs): the simplest FSTs.

Transitions: read one or zero characters, write one or zero characters.

May implement optimal input–output string alignment: “edit distance” (minimal insertions, deletions and substitutions).
Finite-state storage of TU databases/4

Letter transducers (LTs): the simplest FSTs.

Transitions: read one or zero characters, write one or zero characters.

May implement optimal input–output string alignment: “edit distance” (minimal insertions, deletions and substitutions).

Similar TUs likely to share similar alignments: path sharing.
Finite-state storage of TU databases/4

Letter transducers (LTs): the simplest FSTs.

Transitions: read one or zero characters, write one or zero characters.

May implement optimal input–output string alignment: “edit distance” (minimal insertions, deletions and substitutions).

Similar TUs likely to share similar alignments: path sharing.

LT are isomorphic to finite-state automata:
Letter transducers (LTs): the simplest FSTs.

Transitions: read one or zero characters, write one or zero characters.

May implement optimal input–output string alignment: “edit distance” (minimal insertions, deletions and substitutions).

Similar TUs likely to share similar alignments: path sharing.

LT are isomorphic to finite-state automata:

- standard minimization (Hopcroft & Ullman 1979),
Letter transducers (LTs): the simplest FSTs.

Transitions: read one or zero characters, write one or zero characters.

May implement optimal input–output string alignment: “edit distance” (minimal insertions, deletions and substitutions).

Similar TUs likely to share similar alignments: path sharing.

LT are isomorphic to finite-state automata:

- standard minimization (Hopcroft & Ullman 1979),
- lightning-fast implementations (Bentley & Sedgewick 1997).
Finite-state storage of TU databases/5

(‘‘two cases’’, ‘‘dos casos’’)  
(‘‘ten cases’’, ‘‘diez casos’’)

Finite-state storage of TU databases

Letter transducers are easy to maintain: new TUs may be easily added or even deleted while preserving the minimality of the LT! (see our TMI 2002 paper: Garrido-Alenda et al. 2002).
Finite-state storage of TU databases/6

Letter transducers are easy to maintain:
Finite-state storage of TU databases/6

Letter transducers are easy to maintain:

...new TUs may be easily added...
Finite-state storage of TU databases/6

Letter transducers are easy to maintain:

... new TUs may be easily added...

... or even deleted...
Finite-state storage of TU databases/6

Letter transducers are easy to maintain:

... new TUs may be easily added...

... or even deleted...

... while preserving the minimality of the LT!
Finite-state storage of TU databases/6

Letter transducers are easy to maintain:

...new TUs may be easily added...

...or even deleted...

...while preserving the minimality of the LT!

(see our TMI 2002 paper: Garrido-Alenda et al. 2002).
Updating the finite-state translation memory/1
Updating the finite-state translation memory/1

The TM has to be updated using new bitexts.
Updating the finite-state translation memory

The TM has to be updated using new bitexts.

Addition of TUs must be compatible with the LRLM scheme.
Updating the finite-state translation memory/2

A sketch of the updating procedure:

• Start the FST at the beginning of the source text;
• Look for longest input prefix matching the source part of a TU;
• Check the target segment against the output;
• No match? Carefully skip source and/or target words and try again.
Updating the finite-state translation memory/2

A sketch of the updating procedure:
A sketch of the updating procedure:

- Start the FST at the beginning of the source text;
Updating the finite-state translation memory/2

A sketch of the updating procedure:

- Start the FST at the beginning of the source text;
- Look for longest input prefix matching the source part of a TU;
Updating the finite-state translation memory/2

A sketch of the updating procedure:

- Start the FST at the beginning of the source text;
- Look for longest input prefix matching the source part of a TU;
- Check the target segment against the output;
Updating the finite-state translation memory/2

A sketch of the updating procedure:

- Start the FST at the beginning of the source text;
- Look for longest input prefix matching the source part of a TU;
- Check the target segment against the output;
- No match? Carefully skip source and/or target words and try again.
Updating the finite-state translation memory/3

Use heuristics to extend and combine the matched TUs with skipped words.

Note: simultaneous matching of input and output natural to FSTs.
Updating the finite-state translation memory/3

Use heuristics to extend and combine the matched TUs with skipped words.
Updating the finite-state translation memory/3

Use heuristics to extend and combine the matched TUs with skipped words.

Note: simultaneous matching of input and output natural to FSTs.
Bootstrapping:

- Initial contents of FSTs are word-for-word and multiword TUs from bitexts:
  - Marcu (2001), "contiguous alignments": all TUs with words in source segment statistically aligned only with words in target segment and vice versa.
  - Other strategies possible.

Notation:

\[(s,T(s))\]

\(T(s)\) is the translation of \(s\).
Updating the finite-state translation memory/4

Bootstrapping: Initial contents of FSTs are word-for-word and multiword TUs from bitexts:
Updating the finite-state translation memory/4

Bootstrapping: Initial contents of FSTs are word-for-word and multiword TUs from bitexts:

- Marcu (2001), “contiguous alignments”: all TUs with words in source segment statistically aligned only with words in target segment and vice versa.
Updating the finite-state translation memory/4

Bootstrapping: Initial contents of FSTs are word-for-word and multiword TUs from bitexts:

- Marcu (2001), “contiguous alignments”: all TUs with words in source segment statistically aligned only with words in target segment and vice versa.

- Other strategies possible.
Updating the finite-state translation memory/4

Bootstrapping: Initial contents of FSTs are word-for-word and multiword TUs from bitexts:

- Marcu (2001), “contiguous alignments”: all TUs with words in source segment statistically aligned only with words in target segment and vice versa.
- Other strategies possible.

Notation: \((s,T(s))\) translation unit \((T(s)\) is the translation of source segment \(s\)).
Updating the finite-state translation memory/5

A sketch of the bitext alignment strategy with examples.

Source:

\[ \# s_1 s_2 s_3 \ldots \]

Target:

\[ T(s_1) T(s_2) T(s_3) \ldots, \]

\( s_1 s_2 s_3 \ldots \) is a LRLM cover of the input.

The symbol \( \# \) stands for start of sentence, text, etc.
Updating the finite-state translation memory/5

A sketch of the bitext alignment strategy with examples.

Perfect alignment: bitext does not provide new TUs:
Updating the finite-state translation memory/5

A sketch of the bitext alignment strategy with examples.

Perfect alignment: bitext does not provide new TUs:

Source: $\# s_1 s_2 s_3 \ldots$

Target: $\# T(s_1) T(s_2) T(s_3) \ldots$,

$s_1 s_2 s_3 \ldots$ is a LRLM cover of the input.
Updating the finite-state translation memory/5

A sketch of the bitext alignment strategy with examples.

Perfect alignment: bitext does not provide new TUs:

Source:  \# s_1 s_2 s_3 \ldots

Target:  \# T(s_1) T(s_2) T(s_3) \ldots,

s_1 s_2 s_3 \ldots is a LRLM cover of the input.

The symbol \# stands for start of sentence, text, etc.
A sketch of the bitext alignment strategy with examples.

Perfect alignment: bitext does not provide new TUs:

Source: \# s_1 s_2 s_3 \ldots

Target: \# T(s_1) T(s_2) T(s_3) \ldots,

s_1s_2s_3\ldots is a LRLM cover of the input.

The symbol \# stands for start of sentence, text, etc.

Conveniently, \( T(\#) = \# \)
Misalignment: The TM needs updating!
Updating the finite-state translation memory/6

Misalignment: The TM needs updating!

Source: \( \ldots s_k s_{k+1} \ldots \)

Target: \( \ldots T(s_k) \ldots \)

where \( T(s_{k+1}) \) is not a prefix of the target text following \( T(s_k) \).
Updating the finite-state translation memory/6

Misalignment: The TM needs updating!

Source: \( \ldots s_k s_{k+1} \ldots \)

Target: \( \ldots T(s_k) \ldots \),

where \( T(s_{k+1}) \) is not a prefix of the target text following \( T(s_k) \).

Find the longest source \( \hat{s}_k \) after \( s_k \) such that \( T(\hat{s}_k) \) is a target prefix.
Misalignment: The TM needs updating!

Source: \( \ldots s_k \ s_{k+1} \ldots \)

Target: \( \ldots T(s_k) \ldots \)

where \( T(s_{k+1}) \) is not a prefix of the target text following \( T(s_k) \).

Find the longest source \( \hat{s}_k \) after \( s_k \) such that \( T(\hat{s}_k) \) is a target prefix.

Then extend \( T(s_k\hat{s}_k) = T(s_k)T(\hat{s}_k) \), to obtain a LRLM cover of the source, and go on.
Updating the finite-state translation memory/7

When matching and extending matches no longer possible, word skipping necessary:
Updating the finite-state translation memory

When matching and extending matches no longer possible, word skipping necessary:

Source: \( \ldots s_l s'_l s_{l+1} \ldots \)

Target: \( \ldots T(s_l) t'_l T(s_{l+1}) \ldots \),

With \( s'_l \) and \( t'_l \) possibly empty word sequences.
Updating the finite-state translation memory/8

Words skipped only in target text?
Updating the finite-state translation memory

Words skipped only in target text?

Source: \( \ldots \ s_l \ s_{l+1} \ \ldots \)

Target: \( \ldots \ T(s_l) \ t'_l \ T(s_{l+1}) \ \ldots \)
Updating the finite-state translation memory

Words skipped only in target text?

Source: \[\ldots s_l \ s_{l+1} \ \ldots\]

Target: \[\ldots T(s_l) \ t'_l \ T(s_{l+1}) \ \ldots,\]

A possible LRLM cover: \[T(s_l \ s_{l+1}) = T(s_l) \ t'_l \ T(s_{l+1}).\]
Updating the finite-state translation memory/9

Words skipped only in source text?
Words skipped only in source text?

**Source:** \( \ldots s_l \ s_{l+1} \ \ldots \)

**Target:** \( \ldots T(s_l) \ t'_l \ T(s_{l+1}) \ \ldots , \)
Updating the finite-state translation memory/9

Words skipped only in source text?

Source: \( \ldots s_l \, s_{l+1} \, \ldots \)

Target: \( \ldots T(s_l) \, t'_l \, T(s_{l+1}) \, \ldots \),

A possible LRLM cover: \( T(s_l \, s'_l \, s_{l+1}) = T(s_l) \, T(s_{l+1}) \).
Updating the finite-state translation memory/10

But one could examine further before adding new TUs:
Updating the finite-state translation memory/10

But one could examine further before adding new TUs:

For example:

**Source:** \( \ldots \ s_l \ s_{l+1} \ s_{l+2} \ s_{l+3} \ \ldots \)

**Target:** \( \ldots \ T(s_l) \ t'_l \ T(s_{l+1}) \ T(s_{l+3}) \ \ldots, \)
Updating the finite-state translation memory/10

But one could examine further before adding new TUs:

For example:

**Source:** \( \ldots s_l \ s_{l+1} \ s_{l+2} \ s_{l+3} \ \ldots \)

**Target:** \( \ldots T(s_l) \ t'_l \ T(s_{l+1}) \ T(s_{l+3}) \ \ldots \),

If \( t'_l \) is similar to \( T(s_{l+2}) \), this may be a reordering: \( T(s_{l+1} \ s_{l+2}) = t'_l \ T(s_{l+1}) \)
A computationally-feasible approach would explore only a selected set of linguistically-reasonable configurations ("productions").

But also: should we add a TU because of a single misalignment? A possibility: store lists of alternate TUs and counts. Keep only the most frequent TU in main transducer. Modify transducer only if the alternate transduction wins.
A computationally-feasible approach would explore only a selected set of linguistically-reasonable configurations ("productions").
A computationally-feasible approach would explore only a selected set of linguistically-reasonable configurations (“productions”).

But also: should we add a TU because of a single misalignment?
Updating the finite-state translation memory/11

A computationally-feasible approach would explore only a selected set of linguistically-reasonable configurations (“productions”).

But also: should we add a TU because of a single misalignment?

A possibility: store lists of alternate TUs and counts.
Updating the finite-state translation memory/11

A computationally-feasible approach would explore only a selected set of linguistically-reasonable configurations ("productions").

But also: should we add a TU because of a single misalignment?

A possibility: store lists of alternate TUs and counts.

Keep only the most frequent TU in main transducer.
A computationally-feasible approach would explore only a selected set of linguistically-reasonable configurations (“productions”).

But also: should we add a TU because of a single misalignment?

A possibility: store lists of alternate TUs and counts.

Keep only the most frequent TU in main transducer.

Modify transducer only if the alternate transduction wins.
A web-based application/1
A web-based application

Similar to a search engine: e.g., it crawls the web.
A web-based application

Similar to a search engine: e.g., it crawls the web.

FST implementation allows speeds of 100,000 words per second (intensive use expected).
A web-based application/2

Services that would be offered:
A web-based application

Services that would be offered:

**Fast (pre-)translation:** with a client-side alignment and post-edition tool; optional submission of new TUs.
A web-based application/2

Services that would be offered:

Fast (pre-)translation: with a client-side alignment and post-edition tool; optional submission of new TUs.

Translated browsing: with link following (as Altavista or Google).
A web-based application/2

Services that would be offered:

Fast (pre-)translation: with a client-side alignment and post-edition tool; optional submission of new TUs.

Translated browsing: with link following (as Altavista or Google).

A TU lookup interface: for translators (as TransSearch).
A web-based application/2

Services that would be offered:

**Fast (pre-)translation:** with a client-side alignment and post-edition tool; optional submission of new TUs.

**Translated browsing:** with link following (as Altavista or Google).

**A TU lookup interface:** for translators (as TransSearch).

**Bitext contribution:** users may contribute candidate bitexts not covered by the engine.
Concluding remarks

Careful integration of current knowledge:

- web mining for bitexts,
- bitext alignment techniques, and
- finite-state technology,

...together with LRLM segmentation heuristics,

may yield a working web-based translation engine in a few years.

Very desirable for MT-poor language pairs.
Concluding remarks

Careful integration of current knowledge:

• web mining for bitexts,
• bitext alignment techniques, and
• finite-state technology,

together with LRLM segmentation heuristics,

may yield a working web-based translation engine.

Very desirable for MT-poor language pairs.
Concluding remarks

Careful integration of current knowledge:

- web mining for bitexts,
Concluding remarks

Careful integration of current knowledge:

- web mining for bitexts,
- bitext alignment techniques, and
Concluding remarks

Careful integration of current knowledge:

- web mining for bitexts,
- bitext alignment techniques, and
- finite-state technology,
Concluding remarks

Careful integration of current knowledge:

- web mining for bitexts,
- bitext alignment techniques, and
- finite-state technology,

...together with LRLM segmentation heuristics,
Concluding remarks

Careful integration of current knowledge:

- web mining for bitexts,
- bitext alignment techniques, and
- finite-state technology,

...together with LRLM segmentation heuristics,

...may yield a working web-based translation engine in a few years.
Concluding remarks

Careful integration of current knowledge:

- web mining for bitexts,
- bitext alignment techniques, and
- finite-state technology,

together with LRLM segmentation heuristics,

may yield a working web-based translation engine in a few years.

Very desirable for MT-poor language pairs.