It-trim

Francis Tyers, Matthew Marting, Kevin Unhammer

Introduction and background

The Problem: Redundant data

Implementation of

Preprocessing the bilingual

Prefixing the bilingual dictionary

parts

lt_tvim in uso

lt-trim in use

Ending Dictionary Redundancy

Conclusion

Acknowledgements

FST Trimming: Ending Dictionary Redundancy in Apertium

Francis Tyers⁰ Matthew Ma

Matthew Marting¹ Kevin Unhammer²

⁰UiT Norgga árktalaš universitehta Romssa, Norga

ftyers@prompsit.com

¹St. David's School Raleigh, NC.

²Kaldera språkteknologi Stavanger, Noreg

unhammer+apertium@mm.st

27th May 2014

FST's in the Apertium pipeline The Problem: Redundant data

A Solution: Interse

Implementation of

Preprocessing the bilingua dictionary

Prefixing the bilingual lictionary

arts

rsection

-trim in use

ndina Diation

Inding Dictionary
Redundancy

onclusion

Acknowledgements

Introduction and background

Implementation of lt-trim

Ending Dictionary Redundancy

Conclusion

FST's in the Apertium pipeline The Problem: Redundant data

A Solution:

mplementation of

Preprocessing the bilingua dictionary

dictionary

vioving unimieci parts

tersection

t-trim in use

nding Diction

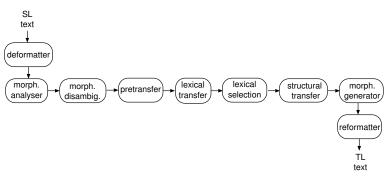
Redundancy

onclusion

- Apertium: Free/Open Source, Rule-based Machine Translation platform
- Goals include:
 - supporting lesser-resourced languages
 - wide coverage
 - post-editable output
 - reusable resources
- Language data (dictionaries, etc.) typically organised in language pairs (Catalan-Spanish, Portuguese-Spanish, etc.)
 - historically: each with its own copy of monolingual data

Apertium pipeline architecture

- Ittoolbox Finite State Transducers used for, among others:
 - morph. analysis: 'fishes' to fish<n><pl>/fish<vblex>>
 - lex. transfer: fish<n><pl> to fisk<n><m><pl>
 - morph. generation: fisk<n><m><pl><def> to 'fiskane'



Francis Tyers, Matthew Marting, Kevin Unhammer

FST's in the Apertium pipeline

FST's in the Apertium pipeline

A Solution: Intersection

Implementation of

Preprocessing the bilingua dictionary

refixing the bilingual ictionary

rts

tersection

t-trim in us

nding Dictionary

onclusion

Acknowledgements

Ittoolbox FST's support a variety of multiwords

An Ittoolbox "lexical unit" is one token, and can be:

- simple non-multi-words: 'fish'
- simple space-separated words: 'hairy frogfish' as a single token
- multiwords with inner inflection: 'takes out',
 analysed as take<vblex><pri><p3><sg># out,
 converted to take# out<vblex><pri><p3><sg> before
 lexical transfer

FST's in the Apertium pipeline

The Problem: Redund

A Solution:

Implementation of

Preprocessing the billingual

Prefixing the bilingual dictionary

loving uninflected le arts

ntersection

t-trim in use

Ending Dictionary Redundancy

onclusion

Acknowledgements

```
pioined multiwords: 'they'll';
analysed as single token
prpers<prn><subj><p3><mf><pl>+will<vaux><inf>,
then split into two tokens
prpers<prn><subj><p3><mf><pl> and
will<vaux><inf> before lexical transfer
```

compounds: 'frogfish';
analysed as single token frog<n><sg>+fish<n><pl>,
then split into two tokens frog<n><sg> and fish<n><pl>
before lexical transfer

FST's in the Apertium pipeline

The Problem: Redunda

Implementation of

Preprocessing the bilingual dictionary

refixing the bilingual ictionary

arts

tersection

t-trim in use

Ending Dictionary Redundancy

Conclusion

Acknowledgements

combinations (space-separated + joined + inner inflection):
 'creure-ho que',
 analysed as single token
 creure<vblex><inf>+ho<prn><enc><p3><nt># que,
 then moved and split into two tokens
 creure# que<vblex><inf> and
 ho<prn><enc><p3><nt> before lexical transfer

The Problem: Redundant data

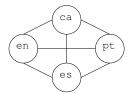


Figure: Ideal number of monodixes with four languages

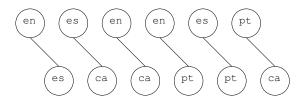


Figure: Current number of monodixes with pairs of four languages

It-trim

Francis Tyers, Matthew Marting, Kevin Unhammer

Introduction and background

FST's in the Apertium pipeline

The Problem: Redundant data

A Solution, intersection

nplementation of

Preprocessing the bilingua dictionary

ctionary

arts tersection

t-trim in use

nding Dictiona

Conclusion

The Problem: Redundant data

A Solution: I

Implementation of

Preprocessing the bilingua dictionary

Prefixing the bilingual dictionary

parts

-t wim in uso

-trim in use

Ending Dictionar

Conclusion

Acknowledgements

Words in analyser but missing from lexical transfer can be problematic:

- 'fishes' to '@fish': loses the inflection
- 'gikk til hundene' "went to the dogs" to 'went to @hund' "went to dog": losing the inflection hides the idiomatic meaning
- ▶ 'öldürmedi' "did not kill" to '@öl' "kill": loses the negation
- lexical transfer is also tag transfer structural transfer thus needs exceptions for half-translated tags

The Problem: Redundant data

But, most importantly, multiword tokenisation means that

'He takes out the trash' translates to 'Han @take out søpla' even though both 'take'-'ta' and 'out'-'ut' are in the bilingual dictionary.

Adding more words makes the translator worse!

rs i s in the Apertium pipe The Problem: Redundant o

A Solution: Intersection

Implementation of

Preprocessing the bilingua lictionary

dictionary

Moving uninflected lemma

loving uninflected arts

ntersection

lt-trim in use

Ending Dictionary Redundancy

Conclusion

Acknowledgements

Compile a *trimmed* analyser-FST containing only entries from monolingual FST that would pass through bilingual FST.

Goal: One big monolingual dictionary, trimmed during compile to language-pair specific analysers.

- We know we can do: FSA1 ∩ FSA2 = FSA3
- We want: output of FST1 ∩ input of FST2 = FST3

The Problem: Redundant da

A Solution: Intersection

Implementation of

Preprocessing the bilingua dictionary

Prefixing the bilingual dictionary

parts

lt=trimi

Endina Diction

Redundancy

Conclusion

Acknowledgements

With some exceptions:

- First need to append .* to FST2
 (because lexical transfer only needs a match on the start of the string)
- And reorder #-multiwords in FST2 (so they look like FST1, otherwise they won't match)
- And let a + in FST1 mean transition-to-start in FST2
 (since single token a+b in FST1 is split into two tokens a b before lexical transfer)

FST's in the Apertium pipeline
The Problem: Redundant data

Implementation of

lt-trim

Preprocessing the bilingua dictionary

refixing the bilingual ictionary

oving uninflected l arts

t-trim in use

Ending Dictionary Redundancy

onclusion

Acknowledgements

Tool takes two compiled FST's, produces a new, trimmed FST

- Preprocess bilingual FST
 - Append .*
 - Reorder #-multiwords
- Depth-first intersection of output of FST1 with input of FST2
 - with an exception on seeing +

Preprocessing the bilingual dictionary

It-trim

Francis Tyers, Matthew Marting, Kevin Unhammer

Introduction and background

FST's in the Apertium pipeline The Problem: Redundant data

implementation o lt-trim

Preprocessing the bilingual dictionary

dictionary

Moving uninflected lemn

ntersection

t-trim in use

Ending Dictionary Redundancy

Conclusion

Prefixing the bilingual dictionary

It-trim

Francis Tyers, Matthew Marting, Kevin Unhammer

Introduction and background

FST's in the Apertium pipeline
The Problem: Redundant data

A SUIULIUII.

Implementation

Preprocessing the bilingua

dictionary

Prefixing the bilingual

dictionary

parts

lt-triminuse

lt-trim in use

Ending Dictionary Redundancy

Conclusion

Moving uninflected lemma parts

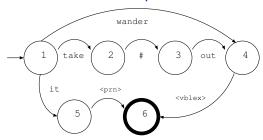
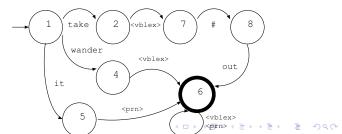


Figure: Input bilingual FST (letter transitions compressed to single arcs)



It-trim

Francis Tyers, Matthew Marting, Kevin Unhammer

Introduction and background

FST's in the Apertium pipeline
The Problem: Redundant data

nplementation of

Preprocessing the bilingua dictionary

Moving uninflected lemma

parts

ntersection

t-trim in use

Ending Dictionary Redundancy

onclusion

Intersection

а

It-trim

Francis Tyers, Matthew Marting, Kevin Unhammer

background background

The Problem A Solution: Intersection

Intersection

Acknowledgenents

Figure: Trimmed monolingual FST

Figure: Input monolingual FST

<n>

<n>

b

<pr>>

<pr>>

6

6

FST's in the Apertium pipeline
The Problem: Redundant data

A Solution: Inte

Implementation o

Preprocessing the bilingu dictionary

Prefixing the bilingual dictionary

arts

lt-trim in use

Ending Dictiona

onclusion

Ending Dictionary Redundancy

It-trim

Francis Tyers, Matthew Marting, Kevin Unhammer

Introduction and background

FST's in the Apertium pipeline
The Problem: Bedundant data

A Solution

Implementation of

Preprocessing the bilingu

Prefixing the bilingual dictionary

Moving uninflected lemr parts

ntersection

lt-trim in use

Ending Dictionary Redundancy

Conclusion

Conclusion

It-trim

Francis Tyers, Matthew Marting, Kevin Unhammer

Introduction and background

FST's in the Apertium pipeline
The Problem: Redundant data

A Solution

Implementation

Preprocessing the bilingu

Prefixing the bilingual dictionary

arts

Intersection

t-trim in use

Ending Dictionary Redundancy

Conclusion

Acknowledgements

It-trim

Francis Tyers, Matthew Marting, Kevin Unhammer

Introduction and background

FST's in the Apertium pipeline The Problem: Redundant data

A Solution

Implementation o

Preprocessing the bilinguidictionary

Prefixing the bilingual dictionary

arts

itersection

t-trim in use

Ending Dictionary Redundancy

conclusion