Workshop on the Generation of Multilingual Parallel Documents

European Commission
Luxembourg, 3 April 2017

M.T. Carrasco Benitez

Three fluid parts

* Touristic tour
* Gory details
* Future directions

Philosophy

* Restate the obvious
* Lie if it helps
* Create practical systems - cost
* Philosophy: Unix - Internet
* Disclaimer: mostly copied

http://www.catb.org/esr/writings/taoup/html
https://www.ietf.org/tao.html

Objectives

* Automatic generation in all languages
* Operation: no human intervention
* Manual vs. industrial production

Scenario 1 - automatic

* Let's dream
* Monthly Data Foo
* Fully automatic generation
* Unix cron - first day of each month
* Multilingual parallel set - parset
* 24 PDF files
* A box with 24 paper publications

Scenario 1 - best case scenario

* Periodic publications
* Lots of text reuse
* Complicated typography
* Lots of prone error data

Example - TEC

* Not new
* 1988
* Travaux en Cours (TEC)
* European Parliament
* Command line interface - CLI
* Standalone system
* Used for several years

Example - CIBA

* Mid 90s
* Common Integrated Budgetary Application (CIBA)
* EU Budget

Applicability

* Not for all publications
* Even 10% very useful
* Typically more
* Often, hard to produce publications

ATP

* Authoring, Translation and Publishing

* Before: preparation - terminology
* After: archiving - data reuse - ready

* Adapt the process to generation

Translation - definition

* Reading -> translate -> writing
* The rest is not translation
* Translators are not typographers
* Translation starts with authoring
* Adapt source text to translation and generation

Translation - dimensions

* Quality
* Speed
* Cost
* Cost = Quality + Speed

* EU: estimation
* 1952 to 2017: 46 billions

http://dragoman.org/cubero

Scenario 2 - typified

* Typified publications
* Types of documents in EUR-Lex
* Example: EU regulations

http://eur-lex.europa.eu

Scenario 2 - reuse

* Text reuse
* Continuum - 0% to 100%
* CIBA: reuse 60% to 85%
* CIBA: best case - 15% new text
* Main work is management and typography
* Real translation is minor

Scenario 2 - comparison

* Scenarios: 1 vs. 2
* Scenario 1: 100%
* Scenario 2: typified - lots of reuse

Scenario 2 - CAA

* Computer-aided authoring
* Text in context - new texts
* Human friendly system
* Layers - similar to cartography
* EU legislation + regulation + agriculture + foo
* Controlled vocabulary - terms - segments

Scenario 2 - CAA - new texts

* Author submits text
* System returns suggestions - segments
* Author accepts/rejects
* Suggestions already translated to all languages
* Similar to on-demand spelling checker

Scenario 2 - CAA - texts/translation

* New texts for the publication
* No new translations
* Source: translation memory silos
* Silos data size: teras or pentas
* Silos: hard for interactive systems

Complexity

* Engineering, not science - MT
* Deceptively simple
* More complex than it looks

Related techniques - monolingual

* From markup to presentation
* troff
* ReSpec
* Bikeshed

https://www.w3.org/respec
https://github.com/tabatkins/bikeshed

Related techniques - merging

* Mail merge
* Template processor
* Data + template = documents
* Often monolingual

Related techniques - DITA

* Darwin Information Typing Architecture

* Inheritance
* Reuse

Related techniques - CMS

* Web content management systems
* Templates

Related techniques - multilingual

* Internationalization - I18N
* Localization - L10N
* Unix environment variables: LANG LC_C*
* Unix commands: locale gettext
* Common Locale Data Repository

CLDR.unicode.org

Simple overview

* Table: es - en - fr - equal
* Template with numbers of the table
* For each language: es - en - fr
* Replace numbers with the segments
* One output file per lang: es - en - fr
* Human computer

Flat level XML

* Entities: &foo; - [foo]

Parstruct

* Pardoc structure
* Needed for real world complex systems
* Contain all items
* Abstract structure

Parstruct - instantiation

* Abstract structure
* Many instantiations
* Reference instantiation - filesystem
* Easy to produce and consume
* Other instantiations: SQLite - filesystem with SQLite - URI - XML
* Structure, how is secondary

Parstruct - URI

* http://es.example.com/foo
* foo: an identifier
* Ouput should be easy to use - low cost
* Human and machine readable

Parstruct - creation

* Creation and maintenance
* Artefacts and data
* Raw editing might be realistic for tabular publications - TEC
* Command line interface - CLI
* Graphical user interface - GUI
* Generated programmatically
* Parstruct is an interface

http://arxiv.org/pdf/0808.3889

Parstruct - levels

* Three levels
* Compromise: complexity vs. cost

[1] parcommon - parallel common - to all

[2] partypes - parallel types - reg

[3] parsets - parallel set - reg 2017/1

Parstruct - example

```
[1] parcommon
    c-definition
    partypes
        [2] regulation
            t-definition
            parsets
                [3] regulation 2017/1
                    s-definition
                    docs: es - en - fr
                [3] regulation 2017/2
        [2] directive
```

Parstruct - components

* Components nodes: parcommon - partypes - parsets

* Abbreviated to "components"

* Components are the roots of the main tree and the subtrees

Definitions - components

* Each component requires a definition

* parcommon: one definition - optional
* partypes: per type - reg, directive
* parsets: per set - reg 2017/1, 2017/2

Definition - structure

* artefact: schema, template, style
* data
  * equal: fix - number - date - ref - quote - synonym - chunk
  * lang: es - en - fr

Definition - artefact

* parstruct -> X-definition -> artefact

* parcommon: entities/vocabularies - include in schema.dtd
* partypes: schema.dtd - reg
* parsets: template.xml – reg 2017/1

Definition - schemas

* parstruct -> t-definition -> artefact -> schema
* It must be adapted to pardoc
* Fine granularity
* No presentation items
* Vocabularies: elements common to related schemas
* Related schemas: regulation - directive
* Vocabulary: Interinstitutional Format Committee (IFC)

http://publications.europa.eu/mdr/ifc

Definition - lang

* parstruct -> X-definition -> data -> lang

* Files with parallel segments
* Files: es - es - fr
* Morphologically independent
* Phrase - multitoken term - token

Definition - equal

* parstruct -> X-definition -> data -> equal
* equal segment for all languages - codes
* A good allied for pardoc
* equal: fix - number - date - ref - quote - synonym
* Keep value as attribute - ISO date

Definition - equal - fix

* Presentation fix: 1000
* XML concatenation: &year;/&serial;
* Year=2017
* Serial=1
* Presentation fix: 2017/1

Definition - equal - L10N

* L10N: localization
* Localizable equal: number - date
* Presentation number: 1.000 - 1,000
* Some transformations possible with XML
* Others more advanced processing

Definition - equal - ref

* References
* Replace code by full reference
* ReSpec
* Normative: [[!RFC3986]]
* Informative: [[RFC3986]]
* Database - specref.org

Definition - equal - quote

* Quotes
* Similar to ref
* Legal texts
* Database

Definition - equal - synonym

* Code to one of several synonyms
* Several methods - random
* Comments for student evaluations

Definition - equal - chunk

* A section or page
* It might contain markups
* Topic - DITA

Run program

* Completed parstruct

* Generate: parstruct -> programs -> output

* Warning about unavailable data
* Guaranteed parallelity

Content output

* Reference canonical markup: clean XML

* Clean XML: simple - no presentation - fine granularity

* Further processing - presentation
* Canonical markups are interfaces

Content output - others

* Other canonical markups
* Procedural: TeX - PostScript - troff
* Descriptive: XML - HTML
* Lightweight: Markdown - wiki
* JSON
* Clean XML could be transformed

Presentation output

* Good compromise: XML + CSS
* Web: HTML + CSS
* PDF
* CSS: Cascading Style Sheets

Further output

* Table of Content - TEC
* Generic approach
* Not specific TEC-like

Grey zone

* In content or presentation
* Automatic numbering - CSS
* equal transformations

http://dragoman.org/laf

Programs

* medinfo
* medrun
* medfoo
* Command line interface - CLI
* Spartan system
* TEC system was simpler
* Used for several years

Libraries

* Transformations
* Equal
* 2017-03-30T16:25:10Z
* Thu 30 Mar 2017 18:25:10 CEST

Package or perish

* Organise files
* Filesystem naming convention
* Filesystem Hierarchy Standard - Linux
* Root directory: foo
* One file: foo.med - zip
* zip: .docx - .odt
* Tree - XML - XLIFF

MED

* Multilingual Electronic Dossier
* Package
* Contains pardoc and the programs

http://dragoman.org/med

Future directions - pargen

* Open source generic system - pargen
* For many types of documents
* Not specific systems: TEC-like
* Specific systems on top pargen

Approach

* Mostly an organisational endeavour
* Not an engineering challenge
* Long-term - forever

Governance

* Foundation

* Copy: Linux, Mozilla, Apache

* Stakeholders: EU - UN - researchers - vendors

Ecosystem

* Copy successful approaches
* Governance and technical
* Internet - IETF

Interoperability

* Internet technologies
* Connection to other systems
* Enterprise Resource Planning (ERP)

Interfaces

* Parstruct
* Canonical markups

Development style

* Running code wins - IETF
* Standard <-> code

Standalone - CLI

* If one cannot implement an existing document in standalone, forget it
* For example: a report from the European Court of Auditors (ECA)
* Later move to new documents
* Mockup regulation
* Real regulation
* Bigger: all the ECA reports - mining

Server

* Cooperation
* Next system: web-based + XML

Many implementations

* At least two independent implementations - IETF

* Generic systems – not TEC-like dedicated system

Email

* Good illustration
* Simpler than pardoc
* Evolving for over half a century

Email - components

* Format: email
* Protocol: SMTP - servers
* Protocols: POP3, IMAP - client

Email - format

* Uses not originally intended
* Mailing lists - HTML

End