Clear3 Issues

Tim Sauerwein / 2020-Dec-29

Table of Contents

[1 Introduction 1](#_Toc60148209)

[2 Code Problems That Should Be Fixed 1](#_Toc60148210)

[2.1 Handling of Integer Overflow in CreatePaths 1](#_Toc60148211)

[2.2 Unqualified Catch Statement in CreatePaths 1](#_Toc60148212)

[2.3 Old Links Handling with Incorrect Assumptions 1](#_Toc60148213)

[3 Clear Algorithm Limitations That Should Be Fixed 2](#_Toc60148214)

[3.1 Assumption that Source Verse Ranges have no Gaps 2](#_Toc60148215)

[3.2 Assumption that Versification Database Entries are in Order 2](#_Toc60148216)

[4 Possible Code Enhancements 2](#_Toc60148217)

[4.1 Improved Datatype for Strong’s Information 2](#_Toc60148218)

[4.2 Parallelization 2](#_Toc60148219)

[4.3 Remove Bare String Uses 3](#_Toc60148220)

[4.4 Added Class for Syntax Tree Node Instead of Bare XElement 3](#_Toc60148221)

[4.5 Reconsidered Secondary Sort in SortPaths() 4](#_Toc60148222)

[5 Resource Service 4](#_Toc60148223)

[6 New Treebanks 4](#_Toc60148224)

[7 Integration with Machine 5](#_Toc60148225)

[8 Integration with the unfoldingWord Environment 5](#_Toc60148226)

[9 Rework of Statistical-Machine-Translation Code 5](#_Toc60148227)

[10 Versification 5](#_Toc60148228)

[11 Segmentation 6](#_Toc60148229)

[12 Persistence 6](#_Toc60148230)

[13 Data Modelling in General 7](#_Toc60148231)

[14 Use of Gateway Translation 7](#_Toc60148232)

[15 Groups 8](#_Toc60148233)

[16 Enhancements to the Tree-Based Auto Alignment Algorithm 9](#_Toc60148234)

# Introduction

This document is my attempt to note and comment on issues in the Clear3 prototype at the moment of writing. Some of these issues are problems to be fixed, and some are areas for possible future development.

# Code Problems That Should Be Fixed

## Handling of Integer Overflow in CreatePaths

In TreeBasedAlignment.CreatePaths the calculation of maxArcs can overflow. The code to handle an overflow should be reconsidered and possibly changed.

## Unqualified Catch Statement in CreatePaths

In TreeBasedAlignment.CreatePaths there is an unqualified catch statement that swallows all possible exceptions. This bad practice should be fixed by either catching specific exceptions or removing the catch statement altogether.

Andi does not remember which exception was in view here; probably we are trying to catch OutOfMemoryException. The Microsoft guidance for catching this exception is to call Environment.FailFast to terminate the app immediately and to add an entry to the system event log.

Probably this catch block needs to be removed. The general approach to memory management probably needs to be at a higher level, and implemented by the client that calls the Clear3 library.

## Old Links Handling with Incorrect Assumptions

In ZoneAlignment.GetMonoLinks, the code that computes existingLinks assumes that the zone consists of a single verse. This assumption is inadequate in general.

A simple fix is not obvious, because the database of old links, from which existingLinks is derived, is organized in a verse-by-verse fashion, and the alternate IDs on which existingLinks is based only make sense within the context of the particular verse from the old alignment in which the alternate IDs occur.

Some thoughtful redesign is needed. The redesign might also need to consider how the larger Clear3 system tackles the versification issues that are before us.

# Clear Algorithm Limitations That Should Be Fixed

## Assumption that Source Verse Ranges have no Gaps

The existing auto-alignment algorithm assumes that the range of source words to be aligned can be described by a starting and ending verse. This assumption is reflected, for example, by the ZoneAlignmentProblem input offered to ZoneAlignment.AlignZone. This assumption is too restrictive for general use (and has already been violated in Charles’s use of Clear2 by an example with non-contiguous source verses).

The ZoneAlignmentProblem input datum is correct in reflecting the current assumptions, but needs redesign to express more general input conditions. This redesign should be pursued as part of considering versification issues in the Clear3 system at large. Tim recommends that the redesigned way of expressing the zone alignment problem should state the source range in terms of the concepts and datatypes for representing versifications, and should continue to accurately reflect the assumptions made by the auto-alignment algorithm.

The current auto-alignment algorithm obtains a syntax tree for the zone by considering the starting and ending verse. The code that constructs this syntax tree is invoked from the TreeService.GetTreeNode method and called from ZoneAlignment.AlignZone. This code in GetTreeNode is also based on the assumption of a contiguous range with a starting and ending verse, and will need rework to become more general. (This code will also need rework if the syntax-tree representations change.)

Tim recommends retaining the current approach of starting from an input specification to obtain first the syntax tree and then the list of SourcePoint objects.

## Assumption that Versification Database Entries are in Order

The implementation of IImportExportService.ImportSimpleVersificationFromLegacy uses an algorithm from Clear2 that assumes that individual records in the legacy versification file format occur in a certain order. It might be worthwhile to remove this assumption and make the algorithm work no matter how the individual records are ordered.

(On the other hand, perhaps the legacy versification file format will be left behind as versification is reconsidered for the Clear3 system at large.)

# Possible Code Enhancements

## Improved Datatype for Strong’s Information

The datatype Dictionary<string, Dictionary<string, int>> that occurs in IImportExportService.BuildStrongTable (and in other places) could be replaced with Dictionary<string, HashSet<string>>.

## Parallelization

It might be worthwhile to parallelize execution in the statistical-machine-translation training and in the tree-based auto-alignment, so as to improve performance by using multiple cores if available.

As part of doing so, the client entry points ISMTService.DefaultSMT and IAutoAlignmentService.AlignZone could be changed to return Task<> datatypes, and could add input parameters of types IProgress<ProgressReport> and CancellationToken.

Note that the current implementation of DefaultSMT already uses the console to report progress.

## Remove Bare String Uses

The Clear3 codebase already goes some way toward using the type system to describe the meanings of strings. For example:

* In the file DataModel\_Alignment.cs the datatype Lemma is defined to represent a string that is to be interpreted as the dictionary form of a source word.
* The SourceID datatype, which identifies the position of a source word in the original manuscript, has a representation as a canonical string. In Clear2 the canonical string form was used exclusively, and the SourceID datatype did not exist. Some, but not all, of these bare string uses have been replaced in Clear3 with use of the SourceID datatype instead.

It might be desirable to go further in the replacement of bare strings with small value types that encapsulate the strings and define their meanings. This strategy makes it easier to understand the code and see that it is correct, and enlists the type system to prevent mistakes.

In particular, Tim suggests considering the following:

* Examine the system at large for uses of bare string, and consider the introduction of new small value types, or the deployment of existing ones, to replace such uses.
* In particular, introduce a new LowerText type for lower-cased target text and change the name of TargetText to TargetSurfaceText, and use the resulting types to distinguish between target lower-cased text and target surface text.
* In particular, look for places where the code is using the canonical strings derived from SourceID, TargetID, and VerseID types, and consider replacement with the types themselves.

## Added Class for Syntax Tree Node Instead of Bare XElement

The Clear3 codebase currently uses the standard .NET type System.Xml.Linq.XElement to represent nodes of the syntax tree.

Tim suggests building on the TreeService to create an object-oriented design for syntax trees and their nodes, in order to:

* encapsulate the particulars of syntax-tree representation away from the rest of the Clear3 system at large,
* provide for an abstract datatype for a syntax-tree node that can be more properly exposed to client code in the SourcePoint datatype, and to
* invent abstractions for syntax trees that can accommodate the variety of different kinds of syntax trees we may wish to work with in future.

## Reconsidered Secondary Sort in SortPaths()

The function TreeBasedAlignment.SortPaths contains a small embedded function named hashCodeOfWordsInPath that is used as a secondary criterion in sorting candidates. Tim left this function in place in the Clear3 prototype in an attempt to match Clear2 behavior. But it might be time to reconsider what this function does.

# Resource Service

* The Clear3 prototype includes a proposed interface for the Resource Service, with just enough of an implementation to execute the regression tests with the default treebank.
* The proposed interface should be reviewed, and the implementation finished.
* In particular, consider:
  + Use of Github to publish resources and as the source for downloading.
  + Rework and/or enhancement of the format of a treebank resource.
  + The possibility of an executable resource associated with segmentation.

# New Treebanks

* Use of alternatives to our current default treebank is certain because of copyright issues.
* New treebanks may have a different format.
  + Randall has already requested and received a statement of what assumptions Clear3 currently makes about the treebank, suggesting a desire to push the boundaries of the format while retaining backward compatibility.
  + The current treebank is structured on a verse-by-verse basis, but Randall has expressed a desire to structure the treebank according to linguistic considerations instead.
* The structure of the treebank is likely to influence these parts of Clear3:
  + The meaning of versification.
  + The construction of zone-pairs according to the versification.
  + The construction of a syntax-tree for a specified zone that covers more than one verse, includes on portions of a verse, and/or has gaps.
  + The generation and test of alignment candidates based on syntax-tree traversal.

# Integration with Machine

* Integration with Machine could require or encourage additional Clear3 interfaces or refactoring of existing interfaces.
* Integration with Machine could require adjustments to the technology stack on which Clear3 is based.

# Integration with the unfoldingWord Environment

* Integration with the unfoldingWord environment seems likely to require porting the Clear3 codebase to TypeScript.
* Some of the datatypes and/or implementation patterns of Clear3 might turn out to be unnatural in TypeScript, and so the port to TypeScript might include some refactoring of Clear3 to keep the C# and TypeScript versions in sync.

# Rework of Statistical-Machine-Translation Code

* The current SMT code in Clear3 allocates a large matrix; if we keep this code, we should consider the use of sparse-matrix techniques to improve performance.
* We have plans to consider replacing the SMT code with FastAlign; on the face of it, such replacement seems straightforward and likely to improve performance, but more work is needed to decide.

# Versification

* The current Clear3 facilities for versification are those of Clear2, which are based on a simple model.
* Our experience with versification suggests:
  + Sometimes it is necessary to define zones that contain more than one verse and/or portions of a verse.
  + The typical translation project does not define its versification carefully.
  + Standard versifications still include diversity in their details.
  + The standard .vrs file does not always express everything that we want to know about the versification.
* Tim recommends:
  + Clear3 should be enhanced to include data structures for expressing a more powerful concept of versification, to be designed by considering lots of examples and attempts by others to do similar things.
  + Clear3 should include operations to import and export versification using the Clear2 legacy XML format and the .vrs file format, perhaps enhanced by an auxiliary file with a new format.
  + Clear3 should include features for checking the versification and diagnosing possible problems.
  + A portion of the versification problem might be well suited to a rule-based approach, perhaps using logic programming.
  + There should only be one versification algorithm in Clear3, not a choice of versification algorithms to be loaded using the Resource Service.
  + The use of gateway translations should be considered as part of working on versification issues.

# Segmentation

* Our opinion is that segmentation is a challenging problem and a possible area for future development.
* Randall believes that segmentation would benefit by taking grammatical information from the syntax trees into account.
* Charles believes that segmentation algorithms should be obtained from the resource manager; this strategy might imply the idea of loading an executable resource at run time.
* The current interface to the segmentation algorithm in Clear3 is based on what the Clear2 segmentation algorithm needed, and is likely not to be general enough for the future. Tim’s suggestion is that a sequence of backward-compatible interfaces for the segmentation algorithm should be developed over time as new segmentation algorithms are investigated and deployed.

# Persistence

* The Clear3 prototype currently supports the legacy Clear2 datatype for persisting alignment. In Clear2 the datatype is called Alignment2, and in Clear3 it is called LegacyPersistentAlignment. Persisting an alignment is realized by reading and writing JSON format based on this datatype.
* Suggested considerations for persistence:
  + A desire for different types of links has already been stated. There will be a need to enhance the auto-alignment algorithm to keep track of why a link was made. Enhancing the Clear3 datatypes named Candidate and TargetBond might be good places to start the thought process.
  + The current datatype for persisting alignment assumes that there are exactly two glosses. This assumption seems restrictive, and might be replaced with the assumption that there are 0 or more glosses. (Note that data for the glosses is a separate input that is used to decorate the persistent datum after alignment is completed.)
  + There are concepts other than alignment that might be appropriate for persistence, such as translation model, versification, gateway translations, and project history. Tim recommends a comprehensive effort to think through persistence in general.

# Data Modelling in General

* The key to creating a set of flexible building blocks that can be reconfigured to meet unanticipated needs and that can be a good basis for future research is a data model that is based on well-defined concepts and that separates concerns.
* Data modelling applies to persistence, to the datatypes that the client exchanges with client entry points, and to the working datatypes that occur internally in the implementation.
* The data used in each Clear3 entry point should be only what that entry point actually uses and produces, without anything extra. Unused data travelling into an entry point leads to technical debt, by improperly coupling entry points together and obscuring the code.
* In particular, the datatypes for persistence should focus just on the issues of persistence, and should not be blindly adopted wholesale for other parts of the system.
* Thinking of the data model in terms of dictionaries, sets, lists, and relations is a good fit to the C# collection and LINQ libraries, and a good way to build on the approach that is already in the Clear3 code (which in turn is a development of Andi’s original point of view).
* The data model should be developed in the context of real workflows and examples; speculative design will almost certainly be wrong, creating code burden and entropy. Speculation belongs in code comments, not the code itself.
* Data modelling should be a central engineering focus based on the best techniques and tools available, not an auxiliary activity or a mere quest for standard formats.

# Use of Gateway Translation

* A gateway translation is an intermediate between the original-language source and the target translation; the source is aligned to the gateway and the gateway is aligned to the target.
* Clear2 has two gateway-translation operations that have not yet been reproduced in Clear3:
  + Perform an auto alignment of source to target, and then express the result as a gateway to target alignment. (Known as “user story 4” in Clear2, with an implementation that calls the AlignG2TviaM2G function in Clear2).
  + Convert a manual alignment of gateway to target into a manual alignment of source to target. (Known as “user story 5” in Clear2, with an implementation that calls the AlignM2TviaM2G function in Clear2).
* These two Clear2 operations are thin layers on top of the main auto-aligner code.
* Clear2 has a datatype named Line3, which is an enhancement of its Line2 datatype for persisting the alignment of one verse, but which also includes a gateway translation. The Line3 datatype holds a manual alignment of source to gateway, and also holds a manual or auto alignment of gateway to target. The Line3 datatype has been reproduced in Clear3 as part of LegacyPersistentAlignmentWithGateway.
* User story 5 in Clear2 converts from Line3 to Line2, and also uses a supplementary auto-alignment to allow certain source to target links for source words that are not linked to the gateway.
* Tim recommends:
  + Evaluate the Clear2 gateway-translation operations in the context of gateway-translation workflows that will actually be delivered to real-world users, as part of specifying gateway-translation operations for Clear3.
  + Include Andi in discussions.
  + To the degree that the Clear2 gateway-translation algorithms apply in Clear3, use the Clear2 code as guidance.
  + Consider refactoring LegacyPersistentAlignmentWithGateway.
  + Make gateway-translation data modelling part of a comprehensive approach to persistence.
  + Make gateway-translation needs part of a comprehensive approach to versification.

# Groups

* Clear2 has rudimentary support for groups, which means some limited consideration of one-to-many, many-to-one, and many-to-many links, as follows:
  + The Alignment2 datatype for persisting an alignment provides for these general kinds of links.
  + Clear2 provides for two inputs with group information:
    - An input file, conventionally called “groups.txt,” stores a mapping of groups of source dictionary forms to possible translations that are each groups of target words with the most important word indicated.
    - When processing the input file of old links (conventionally called “oldAlignment.json”), any link that is not one-to-one results in the update of the groups database (as obtained from “groups.txt”) instead of being added to the old links database.
  + The tree-based auto-alignment algorithm ends with a groups-alignment step that uses a simple algorithm to modify the alignment based on the groups database obtained from “groups.txt” and “oldAlignment.json”. (Note that the groups-alignment step is the only part of the Clear2 auto-alignment algorithm that creates links that are not one-to-one.)
* Clear3 implements some, but not all, of the Clear2 features for groups:
  + The Clear3 LegacyPersistentAlignment datatype is shaped like the Clear2 Alignment2 datatype and so has the same capabilities.
  + Clear3 contains operations for importing from the legacy formats used by “groups.txt” and “oldAlignment.json”, and these operations reproduce Clear2 behaviors.
  + Clear3 **omits** the groups-alignment step in the auto-alignment algorithm. (As a result, the Clear3 auto aligner always produces links that are one-to-one.)
* Andi’s opinion is that the approach to groups in Clear2 is not really the right way to do it, and suggests instead a more comprehensive approach using phrase-based alignment that includes use of the syntax trees.
* The code for the groups-alignment step was ported into Clear3 for safekeeping, and may be found in the file Groups.cs of the Impl.AutoAlign project. This code compiles, but has not been integrated and tested with the rest of the auto-alignment algorithm.
* Tim recommends leaving the Clear2 approach to groups behind, in favor of new features to be developed for phrase-based alignment.

# Enhancements to the Tree-Based Auto Alignment Algorithm

* Ideas for enhancing the tree-based alignment algorithm that Tim thinks he has heard mentioned in conversation include:
  + Andi believes that phrase-based alignment is a key opportunity for tree-based auto-alignment to add value in ways that other approaches cannot.
  + Randall believes that there may be ways to use the syntax trees to assist with segmentation. Tim thinks he has also heard Charles express similar ideas. Tim wonders if such approaches might lead to segmentation becoming part of the alignment process somehow.
  + Tim thinks he remembers a general consensus that more use of grammatical information from the syntax trees would make alignment better.
* Various enhancements and new schemes for syntax trees are being considered by Andi, in Randall’s new trees, and by unfoldingWord. Perhaps these enhancements and new schemes will lead to new ideas in tree-based auto alignment.
* It seems likely that enhancements to the Clear3 tree-based auto-alignment algorithm will require that new ideas and algorithms be added to what is already there. It is therefore very important that the supporting data model be flexible and robust, by separating concerns and using well-defined concepts.