Clear3 Theory of Operation

Tim Sauerwein / 2020-Dec-28

# Introduction

## Purpose and Scope

This document describes the Clear3 codebase at the time of the document’s writing, as a reference for technical staff who maintain and extend the codebase.

## Brief History

### The Clear3 codebase is based on research code written by Andi Wu. Andi’s code is the result of years of study and experiment, and was realized in a codebase that is called Clear2 in this document. The Clear3 codebase is a move from the research environment toward production quality. At present Clear3 implements the Clear2 algorithms without much change.

### One part of the Clear3 codebase trains a statistical-machine-translation model. The original source code was not available. Tim obtained source code for this part by decompiling a DLL. The result resides in the .NET project named **Models**, as further described below.

### The objective of the Clear3 project has been to produced open-source subroutine libraries that are to be linked with client binaries and called in process. The point of view has been to deliver building blocks that the client assembles to support a variety of different client workflows.

### At the moment of writing, the Clear3 codebase has reached a prototype status. It can be used with reasonable confidence for the principal workflow from Clear2. But more work is needed to achieve the overall objectives for Clear3, including:

* Generalization of Clear2 algorithms and additional features, in support of our growing understanding of client workflows.
* Further refinement of the internal design.
* Additional regression tests, unit tests, and sample applications.

# Organization of Project Collateral

## Repository

The Clear3 codebase resides in a Github repository named: gbin-org / Clear3

The repository is organized as follows:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Clear3.sln file at top level | Visual Studio solution for the Clear3 codebase. |
| doc folder | Auxiliary documents (such as the one you are reading now) |
| src folder | Source code except for testing |
| test folder | Source code for testing |

## Solution

The Clear3 codebase is contained in a Visual Studio solution that is organized as follows:

|  |  |
| --- | --- |
| Project | Description |
| Api | Defines interfaces and datatypes for use by the client. (The client should call through the interfaces here, instead of calling code in the Impl projects directly.) |
| Clear3Service | Provides the means for the client to obtain an object that implements the top-level interface defined in Api. |
| Impl.AutoAlign | Implements syntax-tree-based auto-alignment algorithms. |
| Impl.ImportExportService | Implements import from and export to external files, mostly in Clear2 legacy formats. |
| Impl.Miscellaneous | Provides miscellaneous supporting functions to the other Impl projects. |
| Impl.Persistence | Implements creation of objects for persisting client data. |
| Impl.ResourceService | Implements a service for obtaining built-in and downloaded resources (such as syntax treebanks). |
| Impl.Segmenter | Implements algorithms for breaking down a translated text into its linguistic segments. |
| Impl.SMTService | Implements algorithms to compare translated verses to original-language verses by using statistical-machine-translation techniques. |
| Impl.TreeService | Implements algorithms to use syntax trees. |
| Impl.Utility | Implements the utility portion of the client interface. |
| Models | Implements statistical-machine-translation algorithms that were developed at IBM. Was written under Andi’s guidance by a third party from published descriptions of the algorithms. The original source code was not available, so Tim decompiled a DLL to produce the source code that is in our repository now. The code in Models is unchanged from Clear2. |
| RegressionTest1 | A test of the Clear3 codebase that reproduces the principal workflow found in Clear2, and that uses collateral from a Clear2 example for comparison with Clear2 behavior. |
| RegressionTest3 | A test of the Clear3 codebase that uses similar collateral to RegressionTest1, but focuses only on the final tree-based auto-alignment step. |
| SubTasks | A few preliminary attempts to organize some of the Clear3 APIs into higher level patterns. |
| TransModels | A layer of code on top of Models, concerned with using the statistical-machine-translation algorithms, and unchanged from Clear2. |
| UnitTest.Impl.AutoAlign | A few unit tests for some of the code in Impl.AutoAlign. |

# The Principal Clear2 Workflow (as in Regression Test 1)

This section explains the principal workflow of Clear2 and how it is realized in the Clear3 codebase. The explanation proceeds by discussing Regression Test 1. The code may be found in the RegressionTest1.Program.Main method, which is in the Program.cs file of the RegressionTest1 project.

The diagram below summarizes the steps in Regression Test 1, and is followed by some detailed notes on the diagram.

Text

Description automatically generated

***Connect to Clear3 API***: The client calls the static method Clear30Service.FindOrCreate(), which is defined in the Clear3Service project, to obtain an object that implements IClear30ServiceAPI. The interface IClear30ServiceAPI, which is defined in the API project, is the top-level client interface for Clear3, and the means of accessing all of the services that Clear3 provides to its client.

***Get the standard syntax trees***: The syntax trees represent the content and grammar of the original-language manuscripts of the Bible. The client obtains a bank of syntax trees, in the form of an ITreeService object, from the Clear3 Resource Service. The method GetStandardTreeServiceSubtask.Run() packages up the steps to interact with the Resource Service and to download the treebank if necessary. At present there is just one treebank, which is built into the prototype. In future there will be multiple treebanks, and the client must choose an appropriate one.

***Import auxiliary content from the filesystem***: Examples of auxiliary content include a list of segments that are to be considered punctuation, an auxiliary translation model that is to be considered definitive (called the “manTransModel” because in some workflows it comes from manual alignment), and a table of glosses for original-manuscript words. In Regression Test 1 this auxiliary content is imported from the filesystem to imitate what was done in Clear2, because Regression Test 1 is intended for comparison with Clear2 behavior. In general, this auxiliary content could come from a variety of places as appropriate to a given workflow. In particular, some of this auxiliary content could come from the Resource Service.

***Import and segment the translation***: Segmenting means breaking down a translated text into its linguistic segments. The Clear3 prototype has a DefaultSegmenter, which reproduces the segmentation algorithm from Clear2. We believe that segmentation is a challenging area for future work, and perhaps in future the client will obtain a choice of segmenter from the Resource Service.

***Line up translated zones with source zones using the versification***: A zone is the generalization of a verse. Most of the time a zone will be a verse, but sometimes it may be necessary to group several verses together or to split a verse into pieces. A versification is a method for saying how translated verses correspond to original-manuscript verses. Clear2 has a simple method for expressing the versification, which is realized in Clear3 with the SimpleVersification datatype. We think that more sophisticated forms of versification need to be added to Clear3. The result of lining up translated zones and source zones is the Clear3 ParallelCorpora datatype. This step consults the syntax trees to obtain information about the original manuscript.

***Train a statistical model***: The function words are those so identified in the auxiliary collateral mentioned above, and are things like prepositions and articles. All other words are content words. Regression Test 1 filters out the function words from the ParallelCorpora object, leaving only the content words, and sends the result to the SMTService.DefaultSMT method in the Clear3 API. This method is a wrapper for the code in the TransModels and Models projects; this code is unchanged from Clear2. The result of training is an estimated translation model (which is like a dictionary for original-manuscript words in terms of translated words) and an estimated alignment model (which is a one-to-one correspondence between some of the original-manuscript positions and some of the translated positions).

***Perform syntax-tree based alignment***: To prepare for this step, the code converts the ParallelCorpora into a sequence of ZoneAlignmentProblem objects, one for each pair of original-manuscript zone and translated zone that is to be aligned. Then the code walks through the sequence, calling the AlignZone method from the Clear3 API on each zone pair. Finally, the code converts the results from AlignZone into a legacy persistent datum of type LpaLine. The AlignZone method performs syntax-tree-based alignment, and is the heart of the Clear3 codebase and the primary place where Andi invented new algorithms.

***Serialize the standard persistent datum to an output file***: This step produces an output file named “alignment.json” that contains a JSON serialization of the LegacyPersistenAlignment as produced in the previous step.

# The Client API

The client API consists of entry points that the client calls and of datatypes that the client exchanges with the entry points. These entry points and datatypes are defined in the API project, they are all documented there with comments in the code.

The client entry points are all accessible starting from an object of type IClear30ServiceAPI that the client obtains by calling the static method Clear30Service.FindOrCreate that is defined in the Clear3Service project.

The IClear30ServiceAPI interface is partitioned as described by the following table.

|  |  |
| --- | --- |
| **Name** | **Description** |
| ResourceService | Identify, download, and obtain resources such as treebanks, glossaries, sets of stopwards, and segmenters. |
| ImportExportService | Import and export certain Clear3 datatypes from and to the filesystem, mostly using Clear2 legacy file formats. |
| DefaultSegmenter | The default segmenter that packages the Clear2 segmentation algorithm. |
| SMTService | Train a statistical-machine-translation model with zone pairs to produce estimated translation and alignment models. |
| AutoAlignmentService | Perform syntax-tree-based auto-alignment. |
| Persistence | Services for working with persistent data, such as the Clear2 legacy LpaLine datum. |
| Utility | Miscellaneous utility functions, such as filtering function words from a ParallelCorpora object. |

# Principal Datatypes

The most important datatypes in the Clear3 codebase are summarized in the following table. These datatypes are also documented by comments in the code. The middle column of the table notes when the datatype occurs in the client API.

| **Name** | **API** | **Description** |
| --- | --- | --- |
| AlignmentModel | x | A database of possible alignments between SourceID and TargetID, each with an associated score. Can be produced by training a statistical-machine-translation model. |
| Candidate |  | A possible alignment between some of the SourcePoint and TargetPoint objects within a zone; is either a link between one SourcePoint and one TargetPoint, or the certainty that one SourcePoint is linked to nothing, or else the union of two sub-candidates; used in the internals of the syntax-tree-based auto-alignment algorithm. |
| IAutoAlignAssumptions | x | The abstraction for an object that gathers together the auxiliary assumptions used to guide the syntax-tree-based auto-alignment process; the client may supply its own object that implements this interface, or use a standard object created by IAutoAlignmentService. MakeStandardAssumptions. |
| ITreeService | x | An abstract datatype that represents a syntax treebank as obtained from the Resource Service; the syntax-tree-based auto-alignment algorithm converts such an abstract datatype to a concrete datatype internally. |
| LpaLine | x | A Clear2 legacy datatype for persistent alignment data, corresponding to an entry in the Clear2 legacy output JSON format that is stored in a file conventionally named “alignment.json”. |
| MaybeTargetPoint |  | A datum that is either a TargetPoint or is nothing; used in the internals of the syntax-tree-based auto-alignment algorithm. |
| MonoLink | x | An association between one SourcePoint and one TargetPoint with an associated score. |
| ParallelCorpora | x | A list of zone pairs, each of which contains a list of Source objects and a list of Target objects that are to be aligned with each other. |
| SimpleVersification | x | Represents the versification concept from Clear2 consisting of a set of specifications, where each specification indicates a set of source verses that should be associated with a set of target verses. |
| Source | x | A particular instance of a source segment, with its surface text, dictionary form, and position in the original manuscript. |
| SourceID | x | Identifies the position of a segment instance in the original-language source manuscript as known to the syntax tree, in terms of its book, chapter, verse, word, and subsegment numbers. |
| SourcePoint | x | Describes a source segment instance in the context of its membership within a zone that is to be aligned, with descriptive data including things like its dictionary form, its associated node and position in the syntax tree, its position in the original manuscript, and its position within the zone. |
| Target | x | A particular instance of a word in the target translation after segmentation, with its text and its position in the translated text. |
| TargetID | x | Identifies a particular translated word instance as known to the translation by its book, chapter, verse, and word numbers. |
| TargetPoint | x | Describes a target translated word in the context of its membership within a zone that is to be aligned, with descriptive data including things like its text and its position within the zone. |
| TranslationModel | x | A database of possible meanings for dictionary forms of source words; each meaning is the lowercased text of a target word with an associated score. Can be produced by training a statistical-machine-translation model. |
| TreeNodeID |  | A datum that uniquely identifies a node of the syntax tree. |
| TreeNodeStackID |  | A datum that identifies a node of the syntax tree; this identification is unique, except that a node with just one child has the same identification as that child (so that all nodes in a “stack” are identified together). |
| VerseID | x | A datum that identifies a verse of the Bible by its book, chapter, and verse numbers. |
| XElement |  | This type is actually a standard .NET type from the System.Xml.Linq namespace, and is used for syntax-tree nodes. |
| ZoneAlignmentProblem | x | Input data for syntax-tree-based auto-alignment of a single zone, consisting of a list of Target objects and identification of the first and last verses in the original-language manuscript range that is to be matched to the Target objects. |
| ZoneContext | x | Information about the context of a zone to be aligned, consisting of a list of SourcePoint objects and a list of TargetPoint objects for the zone. |
| ZoneMonoAlignment | x | The result of the syntax-tree-based auto-alignment algorithm for a single ZoneAlignmentProblem, consisting of a ZoneContext and a list of MonoLink objects. |

# Call Graph for AlignZone

The implementation of the client API IAutoAlignmentService.AlignZone eventually arrives at the static method ZoneAlignment.AlignZone. This static method is the principal entry point in the ZoneAlignment class, where the method is supported by other static methods in the class and by three principal entry points in three other classes. This topology is shown in the following call graph.

Diagram

Description automatically generated

The static method GetTerminalCandidates is the principal entry point in the class TerminalCandidates, where the method is supported by other static methods of the class, as the following call graph shows.

Diagram

Description automatically generated

The static method AlignTree is the principal entry point in the class TreeBasedAlignment, where the method is supported by other static methods of the class, as the following call graph shows.

Diagram

Description automatically generated

The static method ImproveAlignment is the principal entry point in the class AuxiliaryAlignment, where the method is supported by other static methods of the class, as the following call graph shows.

Diagram

Description automatically generated