Speechcake: Version Control for Speech Corpora

Vlad Dumitru[1], Matthias Boehm[2], Martin Hagmüller[1], Barbara Schuppler[1],
[1]Signal Processing and Speech Communication Laboratory,
Graz University of Technology, Inffeldgasse 16c, 8010 Graz, Austria
[2]Berlin Institute for the Foundations of Learning and Data,
Technische Universität Berlin, Ernst-Reuter-Platz 7, 10587 Berlin, Germany

Correspondence: Martin Hagmüller hagmueller@tugraz.at

Abstract

While the audio recordings of a corpus represent the ground truth, transcriptions are – in the case of manual annotations – subject to human error, and subject to changes related to technology improvements underpinning automated annotation methods. In order to facilitate the dynamic extension of speech corpora, we introduce *Speechcake*, a tool for centralized version control for speech corpora, enabling the automatic check-in and merging of annotations. It considers typical workflows of phoneticians, linguists and speech technologists, and enables the development of dynamic, collaborative, and perpetually-improving speech corpora.

1 Introduction

Speech corpora are generally distributed as static artifacts: after the initial publication, few updated versions are released as snapshots, if any at all. This *one-shot* release mechanism has a negative impact on (1) the organization publishing the corpus, and (2) on the larger research community using the data in question: (1) Collecting, annotating, and packaging a corpus requires a significant investment in terms of time and human effort. Releasing a corpus as a static artifact is done only when the annotation process is *complete*. A dynamic release mechanism allows this effort to be spread over a larger window of time. (2) A static corpus fails to acknowledge that annotations might contain errors and annotator idiosyncrasies. Rosenberg, 2012 highlights some issues found in *classical* static corpora such as the Penn Treebank (Marcus et al., 1993), Switchboard (Godfrey et al., 1992), Hub-4 (Graff et al., 1997) and Boston University Radio News Corpus (Ostendorf et al., 1995), proposing version control software as a solution. The core problem to solve is reproducibility. When researchers correct errors they find in a given dataset, these changes are not propagated back to the original artifact, thereby making it impossible for other parties to reproduce studies resulting from this *locally modified/corrected* corpus.

This paper presents *Speechcake*, a system to provide the necessary tools for creating, extending, and distributing *dynamic* speech corpora. Currently, *Speechcake* supports working with annotations in the Praat TextGrid format[1], although the system can be easily extended to other similar formats (i.e., a collection of tiers made from sequences of time-ordered items). Diverging changes that result from multiple annotators working on the same set of files are resolved through *three-way merging*, the successful result of which contains the changes introduced by two parent versions, relative to a common ancestor.

In practice, our system requires minimal setup to use, either through its built-in web interface, which only requires a relatively modern web browser, or programmatically through its HTTP API. *Speechcake* is built such that it integrates easily in typical workflows of phoneticians and linguists (e.g., manual annotation requiring spectrogram reading), but also in workflows of speech technologists (e.g., automatic speech recognition – ASR – tasks). Overall, *Speechcake* helps to improve the quality and consistency of annotations across several annotation layers, and facilitates the working processes of speech scientists and technologists.

The software package consists of a web server for serving dynamic corpora and a tool for the local administration of repositories. Our code is open source, available at `https://github.com/SPSC-TUGraz/speechcake`, under the terms and conditions of the MIT license. Submitting issues and feature requests is encouraged.

[1]`https://www.fon.hum.uva.nl/praat/manual/TextGrid_file_formats.html`

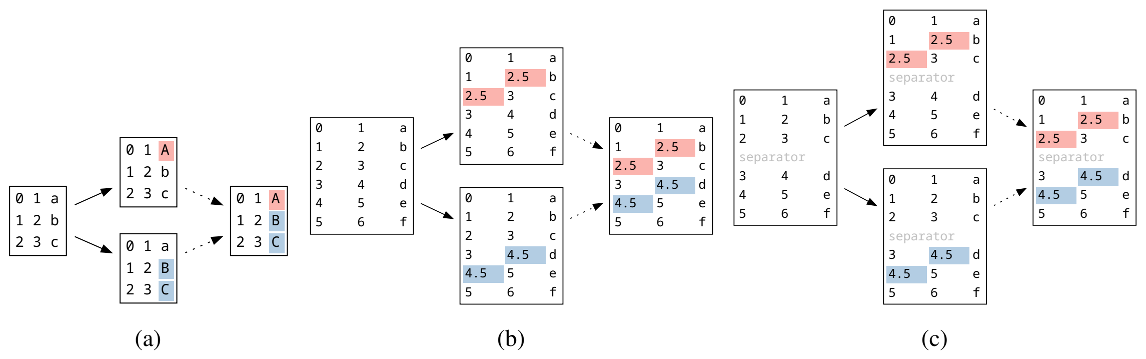

Figure 1: Example merges for common use cases: (a) changing labels of non-overlapping intervals; (b) changing boundaries of non-overlapping intervals; (c) changing boundaries of non-overlapping intervals, with a unique separator added at the separation point between modifications. Line-oriented version control systems, such as for source code management, fail to obtain the merge results in (a) and (b).

2 Related Work

Source Code Version Control In the field of software engineering, source code version control systems make it possible for hundreds or even thousands of collaborators to work on the same set of files. Such systems are inadequate for storing annotation data, since their merge semantics are aimed at resolving conflicts between source code files, as opposed to annotation data. The most commonly used algorithm for merging text files is Diff3, which requires diverging versions to contain the same unique line, such that the modified parts are separated by this common line (Khanna et al., 2007). While not an issue for source code files, since lines such as function definitions or declarations are generally unique within a given file, line-oriented algorithms are a bad fit for annotation data, where a clear separation between modifications may not exist. See for example Figure 1, which shows two scenarios in which line-oriented algorithms cannot merge diverging versions, even when the changed intervals are not overlapping.

Data Version Control As opposed to source code, which can be merged using line-oriented algorithms, data comes in various shapes and sizes, and the merge semantics for one particular type of data may not fit any other. Instead, version control systems focus on particular data formats where merge semantics can be clearly defined. While *Speechcake* is a narrow embodiment where the underlying merge semantics are defined for TextGrid-like annotation data, several other, more general-purpose approaches exist:

Dolt[2] is a database management system that follows the principles of Git, but whereas Git tracks files within a hierarchy, Dolt tracks tables within a database. Dolt databases, much like Git repositories, and *Speechcake* tiers, can be forked, cloned, and merged.

Irmin (Farinier et al., 2015) is an OCaml library that provides the foundation to developing purely functional data structures that can be persisted on disk, merged and synchronized effectively. The library operates on user-supplied data types, which are required to be serializable (for instance, to and from JSON) and mergeable. The merge operation takes two diverging versions and their lowest common ancestor (to be used as the base for the merge). Combinators are provided for typical containers of data, allowing users to declaratively define both the runtime representation of data as well as its merge semantics. In contrast to *Speechcake*, which is a complete solution for version control of annotation data, Irmin is distributed as a library that serves as a foundation for building distributed data stores.

Corpus Management Software Existing speech corpus management systems are not built around the idea of collaborative access. While storing the database itself under version control using an external tool is supported, and even integrated in some of the available solutions, none of them offer automatic reconciliation of diverging versions, which comes as a necessity in the context of multiple annotators working in parallel.

EXMARaLDA is a collection of data formats and software tools for creating, analyzing, and disseminating speech corpora (Schmidt and Wörner, 2009). The software package includes tools for creating and editing transcriptions (Partitur-Editor), creating and managing corpora and their associated meta-

[2]https://github.com/dolthub/dolt

data (CoMa), and querying and analysing corpora (EXAKT). While transcriptions can be individually created and modified by annotators, the software package does not include tools for version control, as *Speechcake* does.

Praaline is an integrated system for managing, annotating, visualising, and analysing speech corpora (Christodoulides, 2018), supporting the most common transcription formats, such as *Praat TextGrid*, *EXMARaLDA*, *ELAN*, and *TranscriberAG*. While the system can be used over the network, also *Praline*, like the earlier mentioned *EXMARaLDA*, does not consider version control and collaborative aspects.

EMU-SDMS is a software package for visualising, annotating, segmenting and querying speech databases (Winkelmann et al., 2017). In Jochim, 2017, the author extends the system with automatic revision control using Git in the background to commit the current state for every modification registered, in a corpus-wide, linear timeline. In comparison, *Speechcake* uses (conceptually) multiple repositories, one for each tier, and allows tiers to be branched and merged individually, without having to align the state of the entire corpus.

Polyglot and Speech Corpus Tools was developed for unified corpus analysis (McAuliffe et al., 2017). The data model uses a graph database for storing annotation graph structures, a relational database for metadata, and a time-series database for acoustic data, combining all three into a *polyglot persistence* solution (Duggan et al., 2015). *Speechcake*, in contrast, covers version control aspects, leaving content-based queries to external tools which can operate on whole snapshots.

3 Data Model of Speechcake

The architecture of *Speechcake* was modelled after the data formats at the boundaries of the system: on the external side, version-control-augmented Praat TextGrid files are used to interact with the outside world. Internally, the annotation structure is that of a TextGrid tier augmented with metadata useful in query processing. On disk, a *Speechcake* repository consists of a single database file containing the entire history of the corpus, allowing for easy backup and maintenance.

Metadata Stamp In a normal *Speechcake* workflow, users check out only a handful of related tiers (annotations of the same primary media). Therefore, in order to be able to trace their origins upon later check-in, tiers need to be augmented (*stamped*) with metadata stored in the names of the tiers, as TextGrid files provide no other opportunity for storing additional information. While this approach preserves compatibility with tools and libraries which interact with TextGrid files, it places a constraint on the users not to impair the integrity of the metadata contained in the stamps.

Annotation Structure As external data model, *Speechcake* uses the TextGrid format for interoperability with other tools. Internally, a *Speechcake document* holds two additional pieces of metadata: a *path* and a *label set*. The path (a logical location) is analogous to the fully-qualified filename where the tier would be stored on a file system (a physical location), relative to the root of the corpus. By decoupling the logical from the physical location, move operations are replaced by simply changing a property on a given tier, thereby offering better feedback for the user as to what changed from one version to the next. The path property, whose length is variable, allows users to organize their corpora in deep, nested structures. The label set can be used for query processing. These labels, as defined by the user for a specific corpus, may contain any kind of information (i.e., speaker IDs, type of annotation, attributes about the recording).

Repository Structure The main purpose of a *Speechcake* repository is to hold a collection of versioned tiers, each identified by a UUID (Leach et al., 2005) assigned upon the tier's initial creation. Tiers can be referenced in two ways: either by their UUIDs, or by their fully-qualified path, which includes the tier name as the terminal component. Note that the former addressing method is immutable (a UUID will always point to the same tier), whereas the latter is mutable. The on-disk format of a *Speechcake* repository contains (1) all past and present versions of tiers, whose contents are split into content-addressable chunks as a form of data de-duplication (Xia et al., 2020), (2) a temporary storage space for tiers that have been checked-in, but not yet submitted to the corpus, and (3) a log of destructively overwriting operations, such as altering the metadata of tiers.

4 Update Process

Check-In Users *check in* a set of locally modified tiers by uploading a TextGrid file via the web interface. Once uploaded, *Speechcake* will first inspect the tiers' metadata and verify that they are part of the corpus. Unknown tiers are rejected, and successfully identified ones go into a temporary storage area, unique to each user. New tiers, identified as such for not having any stamp, can be added to the TextGrid file, and they will be stored alongside the rest of the tiers within the file (i.e., under the same parent path).

Commit Committing a tier involves moving it from the users' temporary area into the corpus, with an optional comment describing the modifications performed. *Speechcake* will then attempt to mark the new version as being the *latest*, which can only be accomplished if the new version has the *current* latest version as a direct parent, meaning that no other modification has been performed since the user has checked out this particular version. If this is not the case, then the newly-submitted version has to be merged with the current latest version, the successful result of which will then be added as another version, and marked as being latest.

Three-Way Merge A merge operation takes two diverging versions X and Y and their lowest common ancestor (or *base* version) B, and produces a new version which incorporates all changes introduced by both X and Y. This mechanism prevents accidental overwriting of data (arising from e.g., two users simultaneously modifying a tier), which may lead to information loss.

For two related tiers A and B, the changes introduced by B on top of A can be described in terms of a *diff* between the two, composed of three sets of items: $A \setminus B$ (items *removed*), $B \setminus A$ (items *added*), and $A \cap B$ (items *kept*). Since tiers are totally ordered sets, where the order operation is given by comparing the lower temporal bound of the two items, the diff operation only needs to iterate over the longer of the two tiers in order to compute the three sets, having an expected-time performance of $\mathcal{O}(\max(|A|, |B|))$. In contrast, the default diff algorithm used in Git has an expected-time performance of $\mathcal{O}(ND)$, where N is the sum of lengths of A and B, and D is the size of the minimum edit script for A and B (Myers, 1986).

To merge X and Y relative to B, *Speechcake* first computes the two diffs $(X \setminus B, B \setminus X, X \cap B)$ and $(Y \setminus B, B \setminus Y, Y \cap B)$. The *common base* C is computed as $X \cap Y \cap B$, and represents the set of items that were unaffected by either versions. Finally, the merged version is obtained by interspersing (by means of set union) both "items added" sets over the common base: $(X \setminus B) \cup (Y \setminus B) \cup C$.

Merge conflicts are detected in the interspersal phase: the union can be computed by iterating over the sets in parallel and moving the item with the lowest lower bound into the output. The algorithm keeps track of the upper bound of the last item copied to the output set, and compares this upper bound with the next incoming lower bound. If the next lower bound comes *before* the last upper bound, then the two items are in conflict, and the operation is aborted.

In order to check whether a merge operation will be successful, the intersection of the sets of *added* items (either points or intervals) must be the empty set, where equality is determined by both the timing and the contents of the item. Otherwise, a merge conflict in the form of a TextGrid file is generated, containing the two conflicting tiers merged except for the conflicting intervals. The user then has adapt the changes, leaving only one tier in the TextGrid file, which upon checking back in is used to resolve the conflict.

5 Conclusions

We introduced *Speechcake*, a version control system for speech corpora, which allows for faster development cycles and better collaboration between annotators and scientists. Our tool primarily supports Praat TextGrid files, making it easy to integrate in workflows which already make use of said file format. Questions such as *Where does this file belong?* or *Is my file the latest version?* are posed and answered by the check-in process, whose role is to minimize user input required to store or update files in the corpus. We have shown the inadequacies of line-oriented merge algorithms, and have proposed a novel, semantics-aware solution. Our tool is extendable through a public API through which automated solutions can interact with the repository. We believe our work inspires further developments in domain-specific version control.

6 Acknowledgements

The work by B. Schuppler was partly funded by an Elise Richter grant (V638 N33) and by grant P-32700, both from the Austrian Science Fund.

References

George Christodoulides. 2018. Praaline: An open-source system for managing, annotating, visualising and analysing speech corpora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics 2018*.

Jennie Duggan, Aaron J. Elmore, Michael Stonebraker, Magda Balazinska, Bill Howe, Jeremy Kepner, Sam Madden, David Maier, Tim Mattson, and Stan Zdonik. 2015. The BigDAWG polystore system. *SIGMOD Record*, 44(2):11–16.

Benjamin Farinier, Thomas Gazagnaire, and Anil Madhavapeddy. 2015. Mergeable persistent data structures. In *Proceedings of Journées Francophones des Langages Applicatifs 2015*, Le Val d'Ajol, France.

J.J. Godfrey, E.C. Holliman, and J. McDaniel. 1992. Switchboard: telephone speech corpus for research and development. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing 1992*, volume 1, pages 517–520 vol.1.

David Graff, Zhibiao Wu, Robert MacIntyre, and Mark Liberman. 1997. The 1996 broadcast news speech and language-model corpus. In *Proceedings of the DARPA Workshop on Spoken Language technology*, pages 11–14.

Markus Jochim. 2017. Extending the EMU Speech Database Management System: Cloud hosting, team collaboration, automatic revision control. In *Proceedings of INTERSPEECH 2017*, pages 813–814.

Sanjeev Khanna, Keshav Kunal, and Benjamin C Pierce. 2007. A formal investigation of diff3. In *Proceedings of the International Conference on Foundations of Software Technology and Theoretical Computer Science 2007*, pages 485–496. Springer.

Paul J. Leach, Michael Mealling, and Rich Salz. 2005. A Universally Unique IDentifier (UUID) URN Namespace. RFC 4122, RFC Editor.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Michael McAuliffe, Elias Stengel-Eskin, Michaela Socolof, and Morgan Sonderegger. 2017. Polyglot and Speech Corpus Tools: A system for representing, integrating, and querying speech corpora. In *Proceedings of INTERSPEECH 2017*.

Eugene W. Myers. 1986. An O(ND) difference algorithm and its variations. *Algorithmica*, 1:251–266.

Mari Ostendorf, Patti J Price, and Stefanie Shattuck-Hufnagel. 1995. The Boston University radio news corpus. *Linguistic Data Consortium*, pages 1–19.

Andrew Rosenberg. 2012. Rethinking the corpus: Moving towards dynamic linguistic resources. volume 2.

Thomas C. Schmidt and Kai Wörner. 2009. EXMARaLDA – creating, analysing and sharing spoken language corpora for pragmatic research. *Pragmatics. Quarterly Publication of the International Pragmatics Association*, 19:565–582.

Raphael Winkelmann, Jonathan Harrington, and Klaus Jänsch. 2017. EMU-SDMS: Advanced speech database management and analysis in R. *Computer Speech & Language*, 45:392–410.

Wen Xia, Xiangyu Zou, Hong Jiang, Yukun Zhou, Chuanyi Liu, Dan Feng, Yu Hua, Yuchong Hu, and Yucheng Zhang. 2020. The design of fast content-defined chunking for data deduplication based storage systems. *IEEE Transactions on Parallel and Distributed Systems*, 31(9):2017–2031.

A Limitations

Annotation Format *Speechcake* currently only supports Praat TextGrid files. We plan on extending our tool in subsequent versions to support other annotation file formats. This can be done in one of two ways – either the new format is a subset of TextGrid, in which case *Speechcake* can convert it without loss of information, or the new format is a superset, in which case the three-way merging algorithm needs to be extended to support new merge semantics. Potential users are encouraged to contact us and describe their use cases.

Large File Storage *Speechcake* does not address the storage of primary media (e.g., audio and/or video recordings), as these are not subject to change throughout the existence of the corpus, and supporting integration with large file storage tools would significantly increase *Speechcake*'s implementation complexity due to the need of supporting potentially multiple protocols (e.g., HTTP, FTP, S3) and authentication/authorization methods. Therefore, the storage and distribution of primary media is left to other tools and systems. In order to match the primary media with their annotations, we suggest using the primary media's filename as a component of the annotations' path.

User Management Again for the purpose of limiting the implementation complexity, *Speechcake* has its own user management system, and updates do not interface with protocols such as LDAP for authentication and/or authorization. User-sensitive information such as name, email, and affiliation are kept in a separate database, and within the corpus database, users are only identified by an opaque UUID. This is done in order to comply with the General Data Protection Regulation (GDPR), such that user information can be removed or altered at any time without impacting the history of the corpus. Other tools such as Git include authorship information (name and email) for every commit, making operations such as changing one's name require a full rewrite of the repository's history.

Number of Concurrent Writers The storage backend of *Speechcake* prohibits more than one user from performing modifications on the corpus at the same time. This limitation is not noticeable in practice, since modifications take on the order of milliseconds to complete, and does not affect users who browse or download parts of the corpus – *Speechcake* supports a virtually unlimited number of read operations at any given time, even when another user is performing modifications, in which case readers will see the last valid snapshot of the corpus.

B Ethical Considerations

The paper does not raise any ethical issues, as no human participants were studied. The corpora used for the development of the tool were datasets already published for academic research prior to this work, and they were collected following the international ethical requirements as suggested by the American Psychological Association.