Data-Driven Parametric Text Normalization: Rapidly Scaling Finite-State Transduction Verbalizers to New Languages

Sandy Ritchie, Eoin Mahon, Kim Heiligenstein, Nikos Bampounis, Daan van Esch, Christian Schallhart, Jonas Fromseier Mortensen, Benoît Brard
Google
{sandyritchie, emahon, kheiligenstein, nbampounis, dvanesch, schallhart, jfmortensen, benoitb}@google.com

Abstract
This paper presents a methodology for rapidly generating FST-based verbalizers for ASR and TTS systems by efficiently sourcing language-specific data. We describe a questionnaire which collects the necessary data to bootstrap the number grammar induction system and parameterize the verbalizer templates described in Ritchie et al. (2019), and a machine-readable data store which allows the data collected through the questionnaire to be supplemented by additional data from other sources. This system allows us to rapidly scale technologies such as ASR and TTS to more languages, including low-resource languages.

Keywords: verbalization, data collection, linguistic typology

1. Introduction

Written-domain text can be challenging for automatic speech recognition (ASR) and text-to-speech (TTS) systems to process due to the presence of *non-standard words* or *semiotic classes* such as numbers, money expressions and measure expressions (Sproat et al., 2001; Taylor, 2009; van Esch and Sproat, 2017). Before the pronunciation of semiotic classes can be determined, they must be verbalized: expanded into word sequences (see Table 1). Verbalizers are usually a key part of TTS (Ebden and Sproat, 2015) and ASR (Sak et al., 2013a; 2013b) text processing systems, as well as various data preparation modules.

Written domain	Spoken domain
35	thirty five
$1.50	one dollar and fifty cents
5cm	five centimeters

Table 1: Verbalization of semiotic classes

Approaches to the verbalization problem have generally not been favorable to low-resource languages. Manually configured systems based on finite-state transducers (FSTs), such as Ebden and Sproat (2015), require linguistic expertise as well as knowledge of an FST compiler such as Thrax (Roark et al., 2012) or Pynini (Gorman, 2016) to set up in a new language. This makes scaling up to a wide range of languages difficult due to the expense of sourcing and onboarding an expert for each language. Recurrent neural networks (RNNs) with finite-state covering grammars (Sproat and Jaitly, 2017; Zhang et al., 2019) or dual encoder classifiers (Gokcen et al., 2019) propose to solve the scaling problem by collecting annotated training examples instead of encoding the verbalizations directly in FST grammars. Such approaches still require a large amount of annotated data, as well as the development of language-specific output filters. For under-resourced languages, the candidate pool of linguistic experts or annotators diminishes, as does the availability of existing resources. This situation poses difficulties for both types of approaches described above.

Various attempts to address this problem have been presented. Gorman and Sproat (2016) use an algorithm which relies on knowledge about the possible factorizations of numbers across languages to induce an FST from a dataset of 300 examples. Gutkin et al. (2016) develop an FST-based system for Bangla by taking an existing Hindi system and translating all of the strings into Bangla. A more general approach has been to identify language-independent characteristics of each semiotic class and to construct templates exposing a limited number of parameters which capture the possible variation across languages (Ritchie et al., 2019). This approach is in the spirit of Bender (2009; 2011; 2016); Sproat (2016) and Ponti et al. (2019), as the template parameters were selected based on insights from linguistic typology, in particular those relating to word order, morphological concord, and other morphosyntactic features of verbalization.

These approaches all assume certain universal constraints on possible verbalizations and a closed set of data points required to specify a particular language. This offers the potential to turn the development of verbalizers into a data-collection problem where the quantity of data needed is tractable for low-resource languages.

In this paper, we discuss the nature of the data required to develop verbalizers for low-resource languages, and address the operational question of how to rapidly and efficiently obtain the data in the context of the unified verbalizer approach outlined by Ritchie et al. (2019). We also discuss some design features of our data store and potential benefits of our method for low resource languages, in particular the possibility of integration with the Unicode Common Locale Data Repository (CLDR) (Unicode, Inc, 2019) for greater collaboration with community members and other stakeholders in the development of high-quality verbalization data.

This paper is structured as follows. Section 2 recaps the unified verbalizers approach. We discuss and exemplify the types of data required in Section 3, and describe our data collection questionnaire as well as other data sources including CLDR in Section 4. Section 5 describes the format in which the data is stored and how verbalizers can be generated from the data. Section 6 outlines the benefits of this approach for low-resource languages.

2. Unified verbalizers

Ritchie et al. (2019) describe a system of verbalizer templates with parameters for language-specific features and sub-templates for the different requirements of ASR and TTS verbalizers. In combination with a number names grammar for conversion of cardinal and ordinal numbers, lexical data for the conversion of other written tokens, and parameters for various features of written tokens and their spoken equivalents, these templates can generate verbalizations for a language without the need for hand-written verbalizers.

2.1. Number names

The most complex aspect of converting non-standard words to their spoken form is the conversion of cardinal and ordinal numbers. We achieve this using a labeled set of 300 examples and an induction algorithm which employs real arithmetic to compute all possible factorizations for numbers up to 999 using addition and multiplication. The algorithm then selects the best parse by analogy with similar parses of smaller numbers (see Ritchie et al. (2019, Section 2.3) for more details). Developers can also manually set certain parameters to handle some more complex aspects of number names systems, such as (weak) vigesimal systems,[1] and 'flop' arithmetic.[2] With an induced number name grammar, the cardinal and ordinal number subcomponents of verbalizations can be handled.

2.2. Verbalizer templates

While cardinal and ordinal numbers exhibit significant variation across languages, the verbalization of other semiotic classes is more constrained and can often be captured using some lexical content and a few parameters relating to the style and ordering of components in the written and spoken domains. For example, in written money tokens, the currency symbol can only precede or follow the numbers, while in decimal numbers like '1.23', the fractional part can typically only be read as a sequence of digits, as in 'one point two three', or a cardinal number, as in 'one point twenty three', or both.

This parametric nature of verbalization makes it a good candidate for templatization. Rather than supporting near-identical hand-written verbalizers for each language, we have developed verbalization templates which offer options to set parameters like 'currency symbol precedes numbers' and 'decimal fractional part is read as a digit sequence'. Since both ASR and TTS verbalizers convert written tokens to their spoken form, we can also share these templates between the two systems with only minor modifications for each use case.

[1]Vigesimal systems use 20 as a base for larger numbers, as in French *quatre-vingts*, 'eighty', lit. 'four twenties'.

[2]Flop arithemetic systems exhibit alternative word orders of addends and multiplicands, e.g. German *einundzwanzig* 'twenty one', lit. 'one and twenty'.

Building on this system, we have developed a suite of data collection and storage methodologies which allow us to capture knowledge of verbalization in a structured and standardized format. In the following sections, we discuss the nature of the data that we need to collect to generate verbalizations, followed by a description of some potential sources for this data, with a particular focus on our questionnaire which covers all the data and parameters required. We then discuss methods for ingesting and storing the data and possibilities for generating FSTs directly from the store.

3. Verbalization data types

Data required for verbalizers can be divided into four major categories: written domain, lexical, morphological and syntactic. We will consider these in turn, using examples from high- and medium-resource languages to illustrate each category. We return to the issue of how to gather this data for low-resource languages in Section 4.

3.1. Written domain data

Written domain data are details about writing conventions in the target language, such as:

- whether the full stop or the comma is used to separate decimal numbers (e.g. '1.2' or '1,2');
- which symbol is used to separate numbers in dates (e.g. '1.2.2020' or '1/2/2020');
- the order of elements in written dates (e.g. DDMMYYYY or MMDDYYYY);
- which currency symbols are used and whether they precede or follow the numbers (e.g. '$21' or '21€');
- common phone number formats, including the number of digits in a block, and the separator used (e.g. '1-800-234-5678' or '07123 456 789').

This information is used to constrain the possible inputs for a verbalizer so that it will only convert written tokens which follow these conventions. This reduces the potential for some written tokens to be classified and verbalized inappropriately. For example, in writing systems like English where decimals are separated by a full stop, powers of ten in big numbers are typically separated by a comma, and vice versa in other languages. If the decimal verbalizer knows it should only convert numbers with a full stop as the separator, a token like '1,234' will not be inappropriately classified and verbalized as a decimal number.

3.2. Lexical data

Lexical data primarily consists of spoken equivalents of written tokens. This includes lists of punctuation and other orthographic symbols, emojis and emoticons, currencies, weekdays, months, time zones, etc. Spoken equivalents of written tokens often exhibit variation of two major types. The first is 'free' variation, where a symbol can be verbalized in more than one way in the same context. In US English for example, the hyphen-minus symbol in negative numbers like '-1' can be read as either 'minus' or 'negative'. The more complex kind of variation is the case in which the same written symbol is verbalized differently depending on the context or type of numeric token in which it is used. For example, the hyphen-minus is verbalized in British English in various other ways in different contexts, as shown in Table 2.

Written domain	Spoken domain
-	**hyphen**
-1	**minus** one
1-2	one **to** two
abc-123.com	abc **dash** one two three dot com
3 - 2 = 1	three **take away** two equals one
1-2-2020	first of February twenty twenty

Table 2: British English verbalizations of hyphen-minus in different contexts

We need to capture all contexts in which multi-functional tokens like hyphen-minus are used in order to convert them appropriately.

3.3. Morphological data

Morphological data can include inherent features of nouns like gender, and inflection of nouns and their dependents for features like number.[3] An example can be found in time verbalizations in Romance languages, where the words for 'hour' and 'minute' have masculine and feminine gender features and exhibit singular/plural number splits. In such cases, some numbers (typically only 1, but also other numbers in some languages like Portuguese) exhibit gender agreement with the hour and minute words, even if the latter are not overtly realized. Examples from Portuguese are shown in (1).

(1) a. *uma (hora) e um (minuto)*
one.F hour.F.SG and one.M minute.M.SG
'one minute past one'

b. *duas (hora-s) e dois (minuto-s)*
two.F hour.F-PL and two.M minute.M-PL
'two minutes past two'

Here the hour and minute words exhibit a singular/plural split (marked by a final *-s* on the plural variants) and the cardinal numbers agree with them in gender. In order to generate these verbalizations, we need to know all the relevant inflectional forms that the hour and minute words can take, as well as those of the numbers that agree with them.
Another common type of morphological marking is case marking. In languages like Russian, verbalizations of some written tokens can exhibit different marking depending on their grammatical function in the clause, like subject, (indirect) object, etc. See Sproat (2010) for a discussion of the problem in Russian number names.

3.4. Syntactic data

Syntactic data primarily includes word order parameters. Nearly all semiotic classes exhibit variation in word order. For example, in classes like decimals, percentages, and temperatures, the words for 'minus', 'percent' and 'degree' can all occur in different orders relative to the numeral. This type of variation can be seen in temperature verbalizations in Malagasy (2a), Bambara (2b) and Swahili (2c), which all exhibit different word orders for the 'minus' and 'degree' words and the numeral.

(2) a. *miiba iray degre*
minus one degree

b. *duguma ni degere kelen ye*
minus degree one

c. *digrii hasi moja*
degree minus one
'minus one degree'

Malagasy follows the word order found in English, with the minus word preceding the numeral and the degree word following. In Bambara, the minus word precedes the degree word, and both precede the numeral. In Swahili, both also precede the numeral, but in this case the degree word precedes the minus word.
Another example of this kind of variation in word order can be found in dates. In date verbalizations, most possible orders of day, month and year are attested across languages, and even within a single language, the order in the spoken form doesn't necessarily match the written order. For example, British English speakers may read a date in DDMMYYYY format (e.g. '1/2/2020') as 'February the first twenty twenty' (month-day-year) instead of 'the first of February twenty twenty' (day-month-year). Both are acceptable readings for this date.

4. Data sources

In the high-resource scenario, recruiting a skilled worker with expertise in a domain-specific formal language like Thrax or Pynini is a viable option for development of verbalizers. They can use a combination of ad-hoc research and consultation with native speakers (or their own intuitions if they are a native speaker) to create custom grammars which produce naturalistic and comprehensive verbalizations for non-standard words.
One potential solution to the verbalization problem for low-resource languages could be to bypass the written domain altogether, and simply transcribe audio training data for ASR in the spoken domain, for example transcribing times as 'two thirty p m' instead of '2:30pm' and so on. However, there are several issues with this kind of approach. First of all, TTS systems would not be able to convert written tokens in any existing text corpora, and our system covers both ASR and TTS. Second, using spoken-to-written and written-to-spoken conversion makes it easier to build natural language understanding and natural language processing systems on top of ASR and TTS systems for low-resource languages, by using existing technology to classify and otherwise operate on semiotic classes like times in the written domain. Finally, and perhaps most importantly, handling high-, medium- and low-resource languages in the same way is more scalable; it is easier to iterate and improve on existing systems if they are all set up in the same way, and this also allows us to share insights and knowledge from higher-resource languages to other languages.

[3]Inflection refers to alternative forms of words depending on their morphological features, for example marking of plural number by the plural suffix *-s* in English 'dollars' versus singular 'dollar'.

This is not to deny the significant issues present in the development of verbalizers for low-resource languages. In general, the resources available in academic literature and online are scarcer, and the pool of potential linguistic consultants with experience in computational linguistics and knowledge of finite-state transducers is significantly smaller. Thus, to produce verbalizations of comparable quality to those in higher-resource languages, we need to turn to other sources of data and linguistic knowledge. The most effective approach appears to be the use of a dedicated questionnaire which asks language consultants with native competence in the target language to transcribe verbalizations of numeric written tokens, as well as answering metalinguistic questions about certain features of these transcriptions.

In addition to such a questionnaire, other potential sources for data include the Unicode Common Locale Data Repository (CLDR), lexical resources like Wiktionary and traditional dictionaries, and typological resources like the World Atlas of Language Structures (WALS) (Dryer and Haspelmath, 2013). Another interesting possibility proposed by Bender et al. (2013) is the use of interlinear glosses in linguistic data to derive lexical and morphosyntactic information. In this section we provide an outline of a questionnaire we have developed for data collection, and briefly discuss CLDR and WALS as potential alternative sources.

4.1. Unified verbalization questionnaire

Verbalization questionnaires have been in use at Google for several years (see Sodimana et al. (2018) for a brief discussion in the context of development of TTS systems for low-resource languages). In tandem with the move to shared verbalizers for TTS and ASR, we significantly expanded our existing questionnaire to cover all types of semiotic classes, taking into account insights from the typology literature and the needs of both ASR and TTS.

The unified verbalization questionnaire takes the form of a Google Sheet with machine-readable labels, which is designed such that educated native speakers with some linguistic knowledge (for example translators or educators) can complete it. It typically asks the respondent to transcribe a representative written token like '-1' and provide alternatives alongside the primary or canonical verbalization. Follow-up questions then target sub-parts of the verbalization in order to discern parameters like word order in the spoken form. For example, alongside questions like "How do you say '-1'?" we include follow-up questions like "Which part of this is '-'?". In this way, we can deduce whether the word for hyphen-minus precedes or follows the number in the spoken form, without the need to ask for this piece of information directly.

This indirect method for eliciting data like word order parameters is inspired by data collection methodologies in linguistic fieldwork, where language consultants sometimes do not have any formal linguistic training, so it is necessary to devise elicitation techniques which access this kind of information through other means than direct questions (e.g. Payne (1997); Bowern (2015), among many others). Here we briefly describe how we use this and other techniques to elicit the different types of data described in Section 3.

We elicit written domain data through various means in our questionnaire, including:

- asking for user input in the case of written symbols such as currency symbols;
- asking the respondent to select from a list in the case of ordering of elements in written dates;
- asking respondents to provide examples in the case of phone number formats.

We then dynamically update follow up questions based on initial responses. For example, if a respondent fills out the currency symbol field with the dollar sign '$', and selects the parameter 'currency symbol precedes number', the follow up questions will then be automatically updated to reflect these facts, for example 'How do you say '$1'?'.

To elicit lexical data, we ask for spoken forms of exemplary written tokens for each class, and also ask respondents to provide alternative forms to capture free variation in the spoken form. For example, to gather information about the verbalization of mathematical expressions, we ask the respondents to transcribe an example like '3 - 2 = 1' with any alternatives. Then in follow up questions, we ask them to identify which parts of the verbalization refer to each symbol, for example "Which part of this is '='?".

Using this approach, we elicit both the mapping between written symbols as well as their verbalizations in specific contexts, which also provides us with test cases with which we can evaluate our template verbalizer. In order to capture context-dependent variation of the type demonstrated in Table 2, the questionnaire includes examples of different written contexts in which multi-functional tokens like hyphen-minus occur. This exposes the overlaps and splits in the use of different words for the same symbol in different contexts.

In order to elicit morphological data, we ask respondents to transcribe verbalizations for representative examples of written currencies, times, and the like, which we have selected to ensure that phenomena like gender features, number splits, and morphological concord will be exposed in the transcriptions. Of course it is not possible to predict exactly what we might find in a new language, but we strive to cover as many features as possible by including singular and plural entities to cover potential singular/plural splits, more than one type of currency to cover potential variation in inherent features like gender among currencies, and examples of numbers in combination with currencies, percentages, degrees and so on to cover potential morphological concord between numbers and the nouns which they modify.

We elicit syntactic data by asking respondents to transcribe at least one representative written token for every relevant semiotic class. In the case of more complex classes like cardinals, ordinals and times, we ask them to transcribe more examples in order to capture the character of the word ordering systems and subsystems for these classes. For example, German exhibits addend flops in numbers 21-99. However, above 100, addends occur in the same order as in English, as demonstrated in (3).

(3) a. *ein -und -zwanzig* (1 + 20)
one and twenty
'twenty one'

b. *zwei hundert eins* (200 + 1)
two hundred one
'two hundred and one'

In order to elicit splits in word order parameters like this, it is necessary to ask for a larger number of examples. This approach to data collection for number names developed by Gorman and Sproat (2016) has led to refinement of our models of morphosyntactic phenomena in several less well-studied languages. For example, we improved our handling of multiplicand flop arithmetic[4] in the Nigerian languages Hausa and Igbo, and noun class agreement phenomena (as well as multiplicand flops) in Bantu languages like Kinyarwanda and Zulu. Again it is not possible to predict what we might find in less well-studied languages, but requesting more examples allows us to capture at least some of the syntactic complexity present in the target language.

4.2. CLDR

For some languages, CLDR also contains a certain subset of the data described in Section 3. It contains some written domain data (e.g. time and date formats) and some lexical data (day, month, time period and era names; days of the week; time zone names; currency names and measure names). For currency and measure names, it also provides information about morphological concord, indicating how the forms differ when quantified by different amounts. This information can be used to extract e.g. singular/plural number splits in currency and measure words. The machine-readable LDML format of this resource makes it amenable to automated ingestion into a common data store.
A drawback of CLDR is the limited number of languages covered. Nearly all of the languages for which they have significant data are already supported in industry applications. Moreover, many languages that have a lot of new and emerging Internet users, such as those in regions of India, Indonesia, and Africa, are not covered by CLDR. CLDR is also limited in some of the information we require for our purposes; for example it lacks symbols and verbalizations for currency subunits (like '25p' $\rightarrow$ 'twenty five pence'). Despite these issues, it is currently the most comprehensive open-source resource for verbalization data. In Section 6., we briefly discuss the potential for expanding the data available in CLDR.

4.3. WALS

WALS offers an interesting source of supplementary support for verbalization data, in particular morphological and syntactic data. For example, the chapter on the order of numeral and noun (Dryer, 2013a) contains parameters for 1153 languages (as of 7th February 2020). For languages where no data is yet available from the questionnaire or CLDR, we could use this information in combination with some basic lexical data to provide best-guess verbalizations for currencies, assuming that the order of the currency word like 'dollar' and the number name follows the stated pattern.
Furthermore, in the case that a language is not listed in that specific chapter, it is also at least theoretically possible to rely on implicational universals of the type introduced by Greenberg (1963) to generate verbalizations. For example, if the chapter on the order of subject, object and verb (Dryer, 2013b) indicates that VSO is the predominant order, even if we do not have information on the order of numeral and noun, it is possible to predict using implicational universals that the numeral will follow the noun. Of course this information will be tenuous at best, but it at least provides some method to localize verbalizations in the absence of any other kind of data.
In this section we identified some possible sources for verbalization data and discussed our data collection methodology via the questionnaire. In the next section we discuss the storage format for the data and briefly touch on the possibility of generating verbalizer FSTs directly from this data store.

5. Data storage and verbalizer generation

While unified verbalizers facilitate verbalizer development, there remains the task of transforming data from different sources into the parameters expected by the templates.
Our solution to this is a data store with a unified data format into which all sources are converted before parameterizing the verbalizers. This means that for any new data source, all that is needed is to convert the data into this format. We use a machine-readable protocol buffer[5] as the data storage format. In its present iteration, the store focuses primarily on lexical (and some basic morphological) data, but our goal is to extend it to all types of data discussed in Section 3.
Entries in the data store take the form of protocol buffers. The data store representation for a particular verbalizable entity provides:

- the data type (e.g. emoji, currency, measure unit, etc.);
- the reference key (typically a written form of the entity);
- possible verbalizations;
- data source;
- time of entry or update;
- any additional entity- and/or language-specific features, such as morphological features like singular/plural number.

Data from each source are imported to the data store when and if they are available, incrementally constructing a gradually more complete set of available data. For example, a record for the verbalization of US dollars in Spanish can be formalized as shown in Figure 1.
In this example, the currency verbalization has been imported from CLDR, while the verbalization for the subunit has been imported from the questionnaire, since CLDR does not provide such information by design. Although the

[4] In multiplicand flop systems, the order of multiplicands is reversed with respect to the order found in English, so the equivalent of e.g. 'two hundred' (2 * 100) is 'hundred two' (100 * 2).

[5] See https://developers.google.com/protocol-buffers

```
currencies {
  key: "USD"
  currency_verbalization {
    text: "dólar estadounidense"
    number: SINGULAR
    source: CLDR
    timestamp: 1578813004
  }
  subcurrency_verbalization {
    text: "centavo"
    number: SINGULAR
    source: QUESTIONNAIRE
    timestamp: 1578839575
  }
}
```

Figure 1: Entry for verbalization of US dollars in Spanish

main currency information was available in the questionnaire data, it was not re-imported, as in this case, it was identical to the CLDR data. If needed, manual edits such as importing additional verbalizations obtained from other sources, or corrections can be applied to the collected data. Having data from questionnaires, CLDR, and any other ad-hoc sources in a single data store makes it easier to parameterize verbalizers automatically. It is possible to set up an automatic data pipeline where data from the various sources are ingested into a common data store, and then used to automatically produce ASR and TTS verbalizers (see Figure 2).

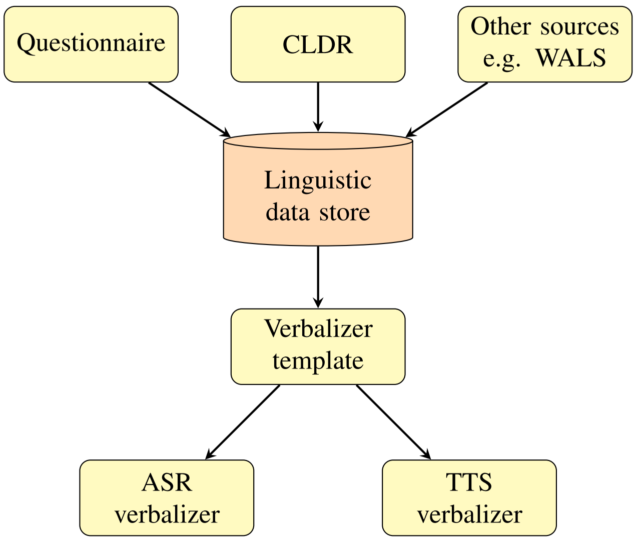

Figure 2: The flow of data from sources to verbalizers

We maintain a source preference hierarchy to reconcile conflicts in the event of competing verbalizations for a given item. All else being equal, a candidate will be selected if its source is preferred according to this hierarchy. For example, a manual correction to a questionnaire entry will eventually take effect and become the default verbalization, since manual entries have been defined as having precedence over any other source. In case of conflicting verbalizations coming from the same source, the newest entry is preferred.

With the above setup, the gradual integration of many sources of data is performed with minimal manual intervention. Having all data in the same place also enables the easy and timely identification of gaps in data coverage. Once all available data have been imported, the data store can be queried for automatic extraction of the necessary data. With further integration of morphological and syntactic information into the store, we can generate verbalization FSTs directly from these specifications, without the need for other intermediate types of representation.

In this section we demonstrated our storage format and preference hierarchy for verbalization data. In the final section we discuss some potential benefits of the system for low resource languages.

6. Benefits for low-resource languages

Producing high-quality verbalizations has long been one of the major bottlenecks in the development of ASR and TTS systems for new languages. Without verbalizers, the quality of text-to-speech and speech-to-text conversion is significantly reduced. Data pipelines like the one we have described in this paper cut down the amount of detailed language-specific work required to set up basic verbalizers for new languages, allowing developers to effectively scale the development of ASR and TTS systems, and to make them available to more language communities faster. Comparing a typical hand-written verbalizer with a configuration file in the new template system, we see a reduction from about 200 lines of custom code to just 25 lines specifying the lexical data, morphological features and syntactic parameters to be used.

A longer-term benefit of our work includes the potential to expand the CLDR data specification based on our analysis of the information needed to set up verbalizers in new languages, so that CLDR can become a central repository of this kind of data. As mentioned above, CLDR already contains a subset of the necessary information, but extending it to cover all relevant information would enable communities and vendors to contribute and share data, paving the way to supporting more languages in more technology platforms. In 2019, we open-sourced part of our number names data for 186 language locales.[6] Open-sourcing verbalization data is more complicated, as it requires further development and refinement of data storage formats and protocols, as well as additional tooling. We are currently working with the CLDR team to explore the options. Our hope is that by including all necessary information in CLDR, everyone would be able to build verbalizers based on data stored in a format which follows well-established protocols from natural language processing as well as insights from linguistic typology.

[6]Verbalizations for numbers 1 to 100 along with powers of ten (1000, 10000, etc.) are available at https://github.com/google/UniNum.

7. Summary

In this paper, we discussed the problem of expansion of written tokens to their spoken form. We recapped Ritchie et al. (2019), in which we discussed the development of shared templates for ASR and TTS verbalizers. We then broke down the types of data required for verbalizers according to their linguistic nature, and discussed our methodologies for data collection using a targeted and typologically informed questionnaire, as well as other supplementary data sources. We then described our data storage format and preference hierarchy for data from different sources. Finally we discussed how our system might help to expand the pool of verbalization data, and showed how through better specification and parameterization of the verbalization problem, we can rapidly scale language technologies such as ASR and TTS to more languages around the world.

8. Bibliographical References

Bender, E. M., Goodman, M. W., Crowgey, J., and Xia, F. (2013). Towards creating precision grammars from interlinear glossed text: Inferring large-scale typological properties. In *LaTeCH@ACL*.

Bender, E. M. (2009). Linguistically naïve != language independent: Why NLP needs linguistic typology. In *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*, pages 26–32.

Bender, E. M. (2011). On achieving and evaluating language-independence in NLP. *Linguistic Issues in Language Technology*, 6(3):1–26.

Bender, E. M. (2016). Linguistic typology in natural language processing. *Linguistic Typology*, 20(3):645–660.

Bowern, C. (2015). *Linguistic fieldwork: A practical guide*. Springer.

Dryer, M. S. and Haspelmath, M. (2013). *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Dryer, M. S. (2013a). Order of numeral and noun. In Matthew S. Dryer et al., editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Dryer, M. S. (2013b). Order of subject, object and verb. In Matthew S. Dryer et al., editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Ebden, P. and Sproat, R. (2015). The Kestrel TTS text normalization system. *Natural Language Engineering*, 21(3):333–353.

Gokcen, A., Zhang, H., and Sproat, R. (2019). Dual encoder classifier models as constraints in neural text normalization. *Proc. Interspeech 2019*, pages 4489–4493.

Gorman, K. and Sproat, R. (2016). Minimally supervised number normalization. *Transactions of the Association for Computational Linguistics*, 4:507–519.

Gorman, K. (2016). Pynini: A Python library for weighted finite-state grammar compilation. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*, pages 75–80.

Greenberg, J. (1963). Some universals of grammar with particular reference to the order of meaningful elements. *In J. Greenberg, ed., Universals of Language. 73-113. Cambridge, MA.*

Gutkin, A., Ha, L., Jansche, M., Pipatsrisawat, K., and Sproat, R. (2016). TTS for low resource languages: A Bangla synthesizer. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2005–2010, Portorož, Slovenia, May. European Language Resources Association (ELRA).

Payne, T. E. (1997). *Describing morphosyntax: A guide for field linguists*. Cambridge University Press.

Ponti, E. M., O'Horan, H., Berzak, Y., Vulić, I., Reichart, R., Poibeau, T., Shutova, E., and Korhonen, A. (2019). Modeling language variation and universals: A survey on typological linguistics for natural language processing. *Computational Linguistics*, 45(3):559–601.

Ritchie, S., Sproat, R., Gorman, K., van Esch, D., Schallhart, C., Bampounis, N., Brard, B., Mortensen, J. F., Holt, M., and Mahon, E. (2019). Unified verbalization for speech recognition & synthesis across languages. In *Proc. Interspeech 2019*, pages 3530–3534.

Roark, B., Sproat, R., Allauzen, C., Riley, M., Sorensen, J., and Tai, T. (2012). The OpenGrm open-source finite-state grammar software libraries. In *Proceedings of the ACL 2012 System Demonstrations*, pages 61–66, Jeju Island, Korea, July. Association for Computational Linguistics.

Sak, H., Beaufays, F., Nakajima, K., and Allauzen, C. (2013a). Language model verbalization for automatic speech recognition. In *ICASSP*, pages 8262–8266.

Sak, H., Beaufays, F., Nakajima, K., and Allauzen, C. (2013b). Written-domain language modeling for automatic speech recognition. In *INTERSPEECH*, pages 675–679.

Sodimana, K., Silva, P. D., Sproat, R., Theeraphol, A., Li, C. F., Gutkin, A., Sarin, S., and Pipatsrisawat, K. (2018). Text normalization for Bangla, Khmer, Nepali, Javanese, Sinhala, and Sundanese TTS systems. In *6th International Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU-2018)*, pages 147–151, 29-31 August 2018, Gurugram, India.

Sproat, R. and Jaitly, N. (2017). An RNN model of text normalization. In *INTERSPEECH*, pages 754–758. Stockholm.

Sproat, R., Black, A. W., Chen, S., Kumar, S., Ostendorf, M., and Richards, C. (2001). Normalization of non-standard words. *Computer Speech Language*, 15(3):287–333.

Sproat, R. (2010). Lightly supervised learning of text normalization: Russian number names. In *2010 IEEE Spoken Language Technology Workshop*, pages 436–441. IEEE.

Sproat, R. (2016). Language typology in speech and language technology. *Linguistic Typology*, 20(3):635–644.

Taylor, P. (2009). *Text-to-Speech Synthesis*. Cambridge University Press.

Unicode, Inc. (2019). Unicode Common Locale Data Repository. `http://cldr.unicode.org/`.

van Esch, D. and Sproat, R. (2017). An expanded taxonomy of semiotic classes for text normalization. In *INTERSPEECH*, pages 4016–4020. Stockholm.

Zhang, H., Sproat, R., Ng, A. H., Stahlberg, F., Peng, X., Gorman, K., and Roark, B. (2019). Neural models of text normalization for speech applications. *Computational Linguistics*, 45(2):293–337.