Proposal for a Unicode Language Inflection Work Group

Last modified: 2024-01-17

Introduction

Inflection is the process of changing the form of a word to express different grammatical features, such as tense, number, gender, or case. In many languages, inflection is a complex and nuanced process, and it can be difficult to implement inflection correctly in software. This can lead to an inability to express native sounding sentences or to errors in text processing, such as incorrect word forms or incorrect grammatical agreement.

For speakers of non-inflected languages, it may be hard to see the importance of inflections. Here is an illustration from Serbian:

① Velike(The big) ② crvene(red) ③ jabuke(apples) ④ su pale(have fallen) sa(from) ⑤ drveta(the/a tree).

- Adjectives ① and ② have to agree in number, gender and case with the noun ③
- Verb ④ (in the past tense) has to agree with the noun ③ in number and gender.
- Noun ⑤ needs the locative case; the nominative would be **drvo**.

The noun cases often correspond to using prepositions in English, so one way to appreciate how bad messages can appear to users in other languages is to consider English sentences with the wrong prepositions and incorrect agreement in number: “There is 3 item of your inbox” vs correct form “There **are** 3 item**s in** your inbox”.

Problem Statement

The problem of inflection is particularly acute in languages that have a large number of inflectional forms, such as all Slavic and Indic languages, Arabic, Korean and Finnish, as well as many other languages, thus affecting a large number of users. In some languages (like Romance languages), inflection affects mostly common words - adjectives, nouns, verbs, but in many languages, proper nouns (like Geo-location names, Brand, People names) can also inflect. Industry so far has either avoided solving this problem or tried solving it for narrow use cases/language combinations. While it is particularly acute in those languages, languages such as French also need to inflect according to gender and number.

LLMs can craft natural-sounding sentences in many languages, including those mentioned above. They have the ability to generate and fill message formats for supported world languages, such as English. Additionally, LLMs can be used to create lexicons and inflection

rules. However, they do have some drawbacks that won't be solved soon - they are large and expensive to train and run, they induce latency for online services and often can't fit on smaller devices without significant quality loss (if at all). They also depend on the quality of data they are trained on, so languages with less (or lower quality) available data may have insufficient quality.

Proposal

The proposal is to form a new Unicode working group to develop a standard for handling inflection in languages. The work group would need to define the scope of the project[1], develop APIs, algorithms and/or ML models for generating inflections[2], and create open source lexicons[3] to support the use of the algorithms and models. We should also allow capability to leverage external/private models and/or lexicons.

The initial goal of the project would be to solve the problem of placeholder replacement, e.g. inflecting dynamic content like a name of a place in the message. Placeholders would be already annotated with necessary grammatical information required of the substitution, so the goal will be to inflect each substitution.

The MessageFormat 2.0 standard proposal already allows for such annotation, such as the following:

English

```
None
.match {$userGender :gender} {$userName} {$count :integer}
{$sourceCity}
* one
{{Hi {$userName}, {$count} package has arrived for you from
{$sourceCity}.}}
* *
{{Hi {$userName}, {$count} packages have arrived for you from
{$sourceCity}.}}
```

English doesn't need the $userGender, so there are no "female" variant messages, and no case value would be needed either (see below).

[1] At this point, we consider the 'unit' to be whatever is in the message; it may have 2 sentences, for example.
[2] The other direction (lemmatization) is also useful, e.g. for indexing and search.
[3] There is a pre-existing Unicode effort to create lexicons, Unilex. We should look at it and extract what's useful.

Italian

```
None
.match {$userGender :gender} {$userName} {$count :integer}
{$sourceCity}
female one
{{Benvenuta {$userName}, {$count} pacco è arrivato per te da
{$sourceCity}.}}
female *
{{Benvenuta {$userName}, {$count} pacchi sono arrivati per te da
{$sourceCity}.}}
* one
{{Benvenuto {$userName}, {$count} pacco è arrivato per te da
{$sourceCity}.}}
* *
{{Benvenuto {$userName}, {$count} pacchi sono arrivati per te da
{$sourceCity}.}}
```

In Italian, “Welcome” has two forms based on $userGender (the gender of the listener/reader): *Benvenuta* for females, *Benvenuto* for males. In fact, the "you" would also change format based on $userGender in languages like Arabic or Hebrew.

Serbian (Latin)

```
None
.match {$userGender :gender} {$userName} {$count :integer}
{$sourceCity}
female one
{{Zdravo {$userName case=vocative}, stigao ti je {$count} paket iz
{$sourceCity case=genitive}}}
…
female *
{{Zdravo {$userName case=vocative}, stigli su ti {$count} paketi
iz {$sourceCity case=genitive}}}
male one
{{Zdravo {$userName case=vocative}, stigao ti je {$count} paket iz
{$sourceCity case=genitive}}}
…
male *
{{Zdravo {$userName case=vocative}, stigli su ti {$count} paketi
iz {$sourceCity case=genitive}}}
```

In Serbian, when inserted for the $sourceCity, “London” (nominative case) becomes “Londona” (genitive case) after “iz” (=”from”) in. Similarly “Petar” (nominative case) becomes “Petre” (vocative case) when being addressed. Serbian also has more than 2 plural forms; those are abbreviated in the examples.

The translation software would expand or contract the match values for the translator's language, as is done now for MF1.0 (aka ICU select format). So the translation software would:

- add female message variants if the language needs them
- add plural variant messages if the language needs them (or remove them) as necessary.

It would also allow the translator to specify the case (case=X), if required by the language.

Beyond supplying the data/code for a module that performs inflection, an associated requirement is to develop the data/code for a module that detects grammatical category values, e.g. noun class (gender, animacy, etc.), of a placeholder, which is often required in order for other placeholders to have grammatical agreement with that first placeholder. The approach to solving this is similar to that of inflections (it can be done via lexicons, rules, LLMs, rules generated by LLMs, etc).

For example, in languages with grammatical gender for objects, in the message “The {$item} costs {$cost}: do you want to buy it?” the gender of $item may cause the gender of “it” to change (to ‘him’ or ‘her’). See also: George Rhoten's comment on the MF2.0 discussion.

Potential approaches

There are a number of potential approaches to solve this problem. Here are some to start the discussion.

- A first idea could be exclusive use of a lexicon for inflection. A lexicon is a database of words and their inflections. Lexicons can be used to look up the correct inflection of a word, and they can also be used to train machine-learning algorithms. Using lexicons alone is not feasible, as they can’t cover all words, e.g. location names, brands etc. and they tend to be large.
- A hybrid approach to the problem of inflection would be a mix of machine learning (ML), lexicons and deterministic algorithms. ML models can derive rules for algorithms and lexicons can take care of exceptions. Lexicons also help provide accurate grammatical information to the models. This approach supports inflection for person names, places etc.
- One of the first things to do (and a relatively simple step) would be to have (or reference) a standardized machine-readable set of terms for grammatical categories. The initial goal would be to flesh out all of the grammatical categories for all of the CLDR languages (about ½ of the languages have data). Those would be keys for the lexicon (curated or ML), and also be useful in MF2.0. We should draw on respected sources for this work.

- We should look at other available grammar processing, such as https://docs.lingoona.com/grammar/ (the description for developers is not very fleshed out; the guide for Authors and translators fills in some of the gaps).

How and what to deliver

Looking at the potential use of the inflection library and data, e.g. in Android and iPhone devices, on web and desktop applications, we should consider how to develop and deploy the solution. One approach is to:

1. Develop code in ICU4C/J/X. This approach has multiple benefits - already set up infrastructure, wide deployment on all platforms, integration with CLDR data, existing contributor pool and WG belonging to ICU TC.
2. Keep the rules and tools that generate those rules within our repository, and only generate needed data for ICU, so they don't have dependency on us. Similar approach can be taken for ML models - training etc would happen on our end.
3. Lexicon data can be spread across CLDR and our repository, and can be used by ICU when deployed. As an added benefit, some CLDR data could be trimmed, as we would be able to generate inflected forms that we now hard code.

Benefit

There are 1.7B[4] Slavic, Arabic and Indic languages native speakers *alone* who are negatively affected by the lack of inflection support in the software. Being able to properly convey information — of professional quality — is a critical part of any UI or user facing application and inflection is a critical part of that ability.

Developing unified inflection solution across industry will help reach those users, and improve localization quality and tooling. Results can also be used by the emerging Unicode MessageFormat 2.0 standard that natively supports inflection, among other features.

Critical User Journeys

Personalization

Addressing a person by name is impossible without proper inflection, e.g. "Hi Petar" in emails has to change from "Zdravo Pet**ar**" → "Zdravo Pet**re**", in UIs on devices, smart assistants etc.

Grammatical Agreement With People

Lots of words can depend on the gender of the audience for a message. For example, saying "Welcome" in Spanish can be "Bienvenido" or "Bienvenida" depending on whether the message

[4] See https://en.wikipedia.org/wiki/Multilingualism_in_India, https://en.wikipedia.org/wiki/Slavic_languages, https://en.wikipedia.org/wiki/Arabic

recipient is masculine or feminine. Another example is in Hebrew and Arabic where you can only say “Here are your messages” when you know the gender of the audience. The possessive second person pronoun attached to the word messages is gendered in those languages. If you don’t know the gender, you can write sentences in an alternative way, but it can sound awkward, passive and less personal. It can be offensive to some cultures when the words chosen do not grammatically agree with the gender identity of the person being referred to.

Travel

Travel sites often present text like “Your flight from Paris to London was delayed.”, where places are dynamically inserted from user selection. One cannot form a natively sounding sentence without inflection.

Improve ICU & CLDR

We support various forms of data in CLDR and expose it through ICU APIs, e.g. standalone month names, measurement units etc. We would improve developer experience if we included inflection feature by removing the need for hard coded forms (standalone, part of the date…).

Timeline

This is a rough timeline: it would be revised based on initial discussions, of course.

Q1 2024 - organize the group, invite people, set up the discussion tools and meetings. Come up with the scope of the effort and define sub-projects, e.g. getting lexicons, defining what infrastructure to use for inflection logic etc.

Q4 2024 - have a few prototypes for X languages, including lexicons to drive them. Outreach to communities, universities etc to contribute.

2025 - stable environment and tools for contributors to add new rules, lexicons.