Jacob Ballou

12/13/2020

Csci 390

Project Assignment 3

**Introduction**

For my class project, I decided to develop an inflector. Inflectors are the opposite of lemmatizers: when provided with a lemma and some other information, an inflector returns the conjugated or declined version of the word. I decided to use a partially rule-based language inflector with a modular design, able to inflect words from a provided dictionary using a set of provided rules. This would allow it to inflect words from any language in the world, as long as the appropriate dictionary and rule set was provided. The idea is, of course, an ambitious one. However, the modular nature of the design makes it easy to test small portions of the code at a time, which made it seem feasible to develop.

I was able to find around ten papers related to grammatical inflection software, which provided me with a sense of how others approached the problem of writing an inflector. In “A Study of Efficiency of Modern Inflection and Lemmatization Software”, for example, the authors came to the conclusion that writing an efficient inflector is more difficult than writing an efficient lemmatizer (Sychev, Gurtovoy, & Penskoy,2017). My experience writing the noun inflector bore this out, although it did not seem to be difficult at first. Only a few of these papers were immediately relevant to the task of writing an inflector. One, “An Extensible Multilingual Open-Source Lemmatizer”, showcases lemmatization software with similar characteristics to my inflector concept (Aker, A., Petrak, J. and Sabbah, F. ,2017).

I decided to implement my inflection software using a modified version of the rule-based inflection detailed in “Open Source Tools for Morphology” (Straková, 2014). I did this in order to reduce the amount of memory space my software took up, while also allowing for a modular configuration and minimizing inaccurate inflections.

While the article “Using the Levenshtein Edit Distance for Automatic Lemmatization: A Case Study for Modern Greek and English”, (Lyras, Sgarbas and Fakotakis, 2007) did not have a direct relation to this project, I did exploit one of the same things the article does: Inflecting a word is equivalent to applying Levenschtein edits. The rules I developed for inflecting words bear a strong resemblance to the rules of editing used in the Levenschtein edit distance. However, my prefix and suffix keywords both correspond to adding a letter, while the replace keyword corresponds to both replacing a letter and removing it. In addition, my rules work with substrings rather than single characters. As such, applying them would be faster than applying the Levenschtein rules, because the rule-applying function would not have to change one character at a time.

In order to test the basic concept of a universal, module-based Inflector, I decided to use JRR Tolkien’s Quenya, a detailed conlang (constructed language) with its own grammatical rules. It was an ideal conlang to use because it is highly inflectional and detailed, with ten noun cases and four grammatical numbers (most languages only have two numbers, singular and plural). Tolkien detailed the correct declension of several Quenya words in a letter to Dick Plotz, allowing for an easy test of the inflection software’s accuracy. Quenya resembles Finnish, Latin, and other highly inflectional languages. As such, two relevant articles were “Declension of Czech Noun Phrases”, by Zuzana Nevěřilová, and “Finnish Noun Inflection”, by Paul Kiparsky. The first details an algorithm to apply a grammatical case to a Czech noun- like this project, but more limited in scope. (Nevěřilová, Z. 2012) The second article was of more general use, because it demonstrates the rules that determine how inflection occurs- the “why” of inflection rather than the “how”. In Finnish, such rules include required patterns of stressed and unstressed syllables. (Kiparsky, P. 2003) Another article, “Predicting Declension Class from Form and Meaning”, was useful because of the information in section 2, Declension Classes in Language. In that section, the authors discuss the ways in which phonetic or phonological factors influence the written form of a declension. (Williams, A., Pimentel, T., McCarthy, A. D., Blix, H., Chodroff, E., & Cotterell, R. 2020)

**Model**

Modeling the problem was straightforward- due to the modular, universal nature of the inflector, I decided to use a nested dictionary to store the rules for each language. Unlike an array, a nested dictionary is not implicitly supported in Python, which makes the structure seem more complicated than it is because I had to create it manually. In this context, a nested dictionary is better than an array because I am using the structure to store information about a language’s tense, gender, number, and so on. Because languages don’t all have the same number of grammatical genders, tenses, numbers, or other characteristics, using a nested dictionary saves space. If a language does not have a certain grammatical gender, the corresponding nested dictionary may safely leave out the entry for that gender.

With an array structure, I would have no such luxury. In fact, each language’s array would be required to have entries for every possible gender, number, tense, and so on. This would obviously take up a great deal of space in memory, most of which would be unused. The more languages I included, the more arrays would be required to hold each language’s rules, so the problem would become even worse the more universal I made the set of inflection modules.

My instructor feedback was that my preliminary results section for Assignment 2 lacked detail in terms of experimentation to validate my proposal. I am unfortunately unable to carry out further experimentation, due to a recurring problem with Jupyter Notebook. I verified that the problem was with Jupyter Notebook and not with my actual code, by writing a much simpler program that encountered the same problem, but ran successfully on my instructor’s computer. As I am unable to carry out further experimentation, I cannot address the feedback.

**Data**

In order to test my rule-based inflector, I needed several examples of properly inflected Quenya nouns. I found two fully declined Quenya nouns provided by J.R.R. Tolkien in a letter to Dick Plotz. The data was not available to download, but I was able to just copy the declined nouns from the Wikipedia entry on Quenya grammar. Since Quenya has ten grammatical cases and four grammatical numbers, each noun has 40 declined forms, providing 80 in total. (Some forms do not exist, lowering this total. Nonetheless, these two nouns provide a wealth of data.)

The contents of the data are reproduced below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Singular** | **Plural** | **Partitive Plural** | **Dual** |
| **Nominative** | Cirya | Ciryar | Ciryalí | Ciryat |
| **Accusative** | Ciryá | Ciryai | Ciryalí | Ciryat |
| **Genitive** | Ciryo | Ciryaron | Ciryalion | Ciryato |
| **Instrumental** | Ciryanen | Ciryainen | Ciryalínen | Ciryanten |
| **Allative** | Ciryanna | Ciryannar | Ciryalinnar | Ciryanta |
| **Dative** | Ciryan | Ciryain | Ciryalin | Ciryant |
| **Locative** | Ciryassë | Ciryassen | Ciryalissen | Ciryatsë |
| **Short Locative** | Ciryas | Ciryais | Ciryalis | **Does not exist** |
| **Ablative** | Ciryallo | Ciryallon | Ciryalillon | Ciryalto |
| **Adjectival** | Ciryava | **Does not exist** | Ciryalíva | **Does not exist** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Singular** | **Plural** | **Partitive Plural** | **Dual** |
| **Nominative** | Lassë | Lassí | Lasselí | Lasset |
| **Accusative** | Lassé | Lassí | Lasselí | Lasset |
| **Genitive** | Lassëo | Lassion | Lasselion | Lasseto |
| **Instrumental** | Lassenen | Lassínen | Lasselínen | Lassenten |
| **Allative** | Lassenna | Lassennar | Lasselennar | Lassenta |
| **Dative** | Lassen | Lassin | Lasselin | Lassent |
| **Locative** | Lassessë | Lassessen | Lasselissen | Lassetsë |
| **Short Locative** | Lasses | Lassis | Lasselis | **Does not exist** |
| **Ablative** | Lassello | Lassellon | Lasselillon | Lasselto |
| **Adjectival** | Lasseva | **Does not exist** | Lasselíva | **Does not exist** |

I chose this data because it is one of the only examples of fully inflected Quenya nouns, and therefore allows very thorough testing of my inflector. Since I obtained the data through the Wikipedia article on Quenya grammar, I did not need to extract any features. While the Wikipedia article in question provides other examples of Quenya noun declension, Tolkien published their declensions several decades before the Plotz letter. As such, they follow the rules of an earlier version of Quenya and would skew any analysis of my inflector’s accuracy.

**Analysis**

In order to determine my inflector’s accuracy, I decided to provide it with the nouns Tolkien used as examples of Quenya declension in the Plotz letter, then compare its output to the correct declensions Tolkien provided. I planned to use cosine similarity, word similarity, and residual sum of squares to provide a meaningful way of determining the accuracy of my code. I would get the cosine-based word similarity between my program’s output and the correct declension to determine similarity for each word. If I subtracted each such similarity from the number one, I would be able to use the results, which would then measure how dissimilar the output and the correct declension were, like the error values in linear regression. I would calculate the RSS using the word dissimilarities, and it would provide a measure of how accurate my program actually is.

Where is the inflected noun created by my program and is the inflected noun Tolkien used in the Plotz letter.

I began testing by trying the rule interpretation function on its own. I provided both the noun and a very simple rule to apply to the noun. However, the rule interpretation function did not return anything, nor did it run at all. Further investigation proved that the problem was with Jupyter Notebook, not my code. Due to this, I was unable to retrieve any data from my program. I was unable to test the nested dictionary structure due to the same issue. However, nested dictionaries are workable in python, according to GeeksforGeeks and many other coding websites. (Semwal, 2020)

**Conclusion**

I was unable to retrieve any data from my program to draw a conclusion with. However, I found proof that my basic approach is workable: Nested dictionaries can be implemented in Python, and “Open Source Tools for Morphology” (Straková 2014) shows that rule-based inflectors are viable. Modular, open-source lemmatizers can be implemented, and a lemmatizer is fairly similar to an inflector (Aker, A., Petrak, J. and Sabbah, F., 2017). I am confident that if it weren’t for the issue with Jupyter Notebook, I would have been able to successfully test my code and obtain usable data.

**Reflection**

During my work on this project, I learned that knowing the theory behind implementing a complex project is not the same thing as actually succeeding in the implementation. I also learned that for shorter, single-person projects, a simpler project is better. Professor Ghosh pointed out that everyone’s projects were overly complicated.

After I submitted the first assignment, I learned that I should be more detailed in my project proposals, and that I needed to be clearer on what the proposal actually was. After I submitted the second assignment, I learned that I also needed to be more detailed when describing my testing procedures.

The main challenge I faced was the issue with Jupyter Notebook. I was unable to overcome this challenge, although I was able to prove that the issue was not with my code. Test code that failed when I ran it succeeded when Prof. Ghosh ran it on his machine. I did learn that when working on a complicated project without the benefit of group coding, the smallest detail can cripple development.

If I had another six months to work on this project, I would, of course, attempt to either fix the issue with Jupyter Notebook or move the project to another platform. After that, I would finish testing my code, importing the data I planned to use, and making sure the results made sense. If I had time after that, I might try to implement machine learning as Prof. Ghosh suggested.

If I had to redo the project, I’m not sure if I would do anything differently. It seems as though the root cause of my problems was an unforeseeable issue with the software I was using to write and test my code. If it weren’t for this issue, I would have been able to continue coding, testing, and refining my program. I would have been able to import the data, compare it to the results produced by my inflector, and draw a meaningful conclusion. As it is, the code I wrote is free of syntax errors, but entirely untested. It should be treated as pseudocode rather than run as-is.

Works Cited:

Sychev, O., Gurtovoy, V., & Penskoy, N. (2017). A Study of Efficiency of Modern Inflection and Lemmatization Software. *Proceedings of the IV International Research Conference "Information Technologies in Science, Management, Social Sphere and Medicine"(ITSMSSM 2017)*. doi:10.2991/itsmssm-17.2017.30

Straková, J., Straka, M., &amp; Hajič, J. (2014). Open-Source Tools for Morphology, Lemmatization, POS ... Retrieved October 20, 2020, from <https://core.ac.uk/download/pdf/48906950.pdf>

D. P. Lyras, K. N. Sgarbas and N. D. Fakotakis, "Using the Levenshtein Edit Distance for Automatic Lemmatization: A Case Study for Modern Greek and English," 19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007), Patras, 2007, pp. 428-435, doi: 10.1109/ICTAI.2007.41.

Williams, A., Pimentel, T., McCarthy, A. D., Blix, H., Chodroff, E., & Cotterell, R. (2020). *Predicting Declension Class from Form and Meaning* (Unpublished doctoral dissertation). Cornell University. Retrieved November 15, 2020, from https://arxiv.org/abs/2005.00626

Nevěřilová, Z. (2012). *Declension of Czech Noun Phrases* (Unpublished doctoral dissertation). Masaryk University. Retrieved November 15, 2020, from https://www.researchgate.net/profile/Zuzana\_Neverilova/publication/301887499\_Declension\_of\_Czech\_Noun\_Phrases/links/572aee9d08aef5d48d30d55d/Declension-of-Czech-Noun-Phrases.pdf

Kiparsky, P. (2003). Finnish Noun Inflection. Retrieved November 15, 2020, from <https://www.researchgate.net/profile/Paul_Kiparsky/publication/265756560_Finnish_Noun_Inflection/links/56bfaa1408ae44da37fa7053/Finnish-Noun-Inflection.pdf>.

Ingason A.K., Helgadóttir S., Loftsson H., Rögnvaldsson E. (2008) A Mixed Method Lemmatization Algorithm Using a Hierarchy of Linguistic Identities (HOLI). In: Nordström B., Ranta A. (eds) Advances in Natural Language Processing. GoTAL 2008. Lecture Notes in Computer Science, vol 5221. Springer, Berlin, Heidelberg. <https://doi.org/10.1007/978-3-540-85287-2_20>

[Makarov, Peter](javascript:contributorCitation(%20'Makarov,%20Peter'%20);); [Clematide, Simon](javascript:contributorCitation(%20'Clematide,%20Simon'%20);) (2018). Neural Transition-based String Transduction for Limited-Resource Setting in Morphology. In: Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, New Mexico, USA, 20 August 2018 - 26 August 2018, 83-93.

Tuomo Korenius, Jorma Laurikkala, Kalervo Järvelin, and Martti Juhola. 2004. Stemming and lemmatization in the clustering of finnish text documents. In Proceedings of the thirteenth ACM international conference on Information and knowledge management (CIKM '04). Association for Computing Machinery, New York, NY, USA, 625–633. DOI:https://doi.org/10.1145/1031171.1031285

Aker, A., Petrak, J. and Sabbah, F. (2017) An extensible multilingual open source lemmatizer. In: Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017. International Conference Recent Advances in Natural Language Processing, 02-08 Sep 2017, Varna, Bulgaria. ACL , pp. 40-45.

Semwal, S. D. (2020, December 03). Python Nested Dictionary. Retrieved December 13, 2020, from https://www.geeksforgeeks.org/python-nested-dictionary/