WHAT LED US TO THE DATA

ZACHARY RYAN
University of Colorado Boulder

The usefulness of Bibles in Natural Language Processing is wildly underrated, especially when looking at low-resource languages. The reason these texts are so useful in machine translation is because it provides parallel text in languages that would otherwise have non or very little other parallel texts available. The question is then, how can we turn these parallel texts into a useable, organized, and reliable data set? There are two general methods that can be used to collect and organize this data, the old fashioned way by hand or automating the process through a computer. While this paper will touch on both methods, it takes a much deeper dive in the automated processing side and looks at some of the reasons this route was chosen.

Keywords: data collection, data organization, machine learning, natural language processing, automation

1. INTRODUCTION

Many researchers and companies have ideas for machine learning or artificial intelligence (AI) projects which often times hit a barrier immediately. Where do I get enough data to create my model? This is the very question myself and two other researchers had to ask ourselves when working with what we consider a LOW-RESOURCE LANGUAGE. For those not familiar with the term low-resource language, it is a language that lacks significant monolingual or parallel corpora that usually requires manually crafted linguistic resources needed for the study of the language. The languages the lead of my team chose to study were Basque and Navajo, which we wanted to use to create working translation models to English from the respective language and vice versa.

1.1. THE TEAM AND ITS RESEARCH

The three of us started working together to because the leader of our team, Mans Hulden, a linguistics professor at University of Colorado, wanted to see how useful Bibles could be for neural machine translation (NMT) models. And if any tokenization methods on top of this could help improve the NMT models. The second member of the team, Ling Liu, a master's student at University of Colorado, was already helping Mans with the linguistic side of the project before I was brought in. I was the last person brought in to help more with the coding side of the project

because I had done some previous work in natural language processing (NLP) with Mans and was also at the time a senior finishing my computer science degree.

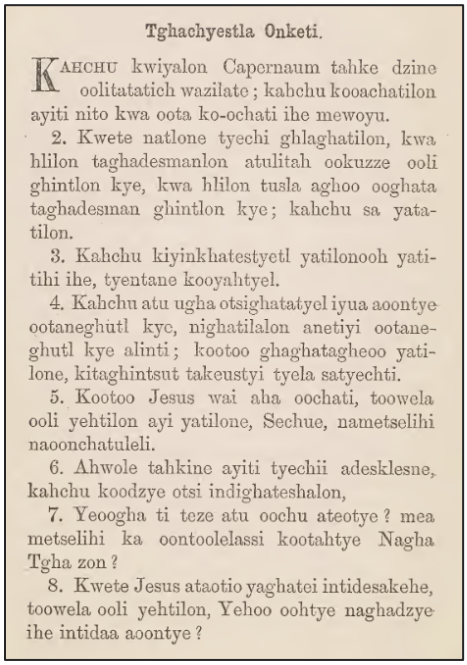
2. CHOOSING A METHOD FOR CREATING TRANSLATION MODELS

To create this type of translation model would be possible through two methods. Large amounts of monolingual text which can be looked at for each language where patterns are found until small accurate translations are made which are then expounded upon. This will eventually lead to very accurate translation models but this has only been shown to be successful with languages with large amounts of text like English and Spanish (Conneau et al. 2018). The second way to create a successful translation model is to use sets of parallel text from each language. Since the translation is known, the model guesses and we tell it if its right or wrong and to adjust from there. This method does not require as nearly as much text as the first option. Since we were working with low-recourse languages we chose the second option to create our translation models.

3. FINDING DATA

The next steps were to now find where we could collect data from. One of us had found a pdf file of pictures of bible scripture written in Navajo. An example of what one of these pages looks like can be seen below in Figure 1.

FIGURE 1



Tghachyestla Onketi.

KAHCHU kwiyalon Capernaum tahke dzine oolitatatich wazilate; kahchu kooachatilon ayiti nito kwa oota ko-ochati ihe mewoyu.

2. Kwete natlone tyechi ghlaghatilon, kwa hlilon taghadesmanlon atulitah ookuzze ooli ghintlon kye, kwa hlilon tusla aghoo ooghata taghadesman ghintlon kye; kahchu sa yatatilon.

3. Kahchu kiyinkhatestyetl yatilonooh yatitihi ihe, tyentane kooyahtyel.

4. Kahchu atu ugha otsighatatyel iyua aoontye ootaneghutl kye, nighatilalon anetiyi ootaneghutl kye alinti; kootoo ghaghatagheoo yatilone, kitaghintsut takeustyi tyela satyechti.

5. Kootoo Jesus wai aha oochati, toowela ooli yehtilon ayi yatilone, Sechue, nametselihi naoonchatuleli.

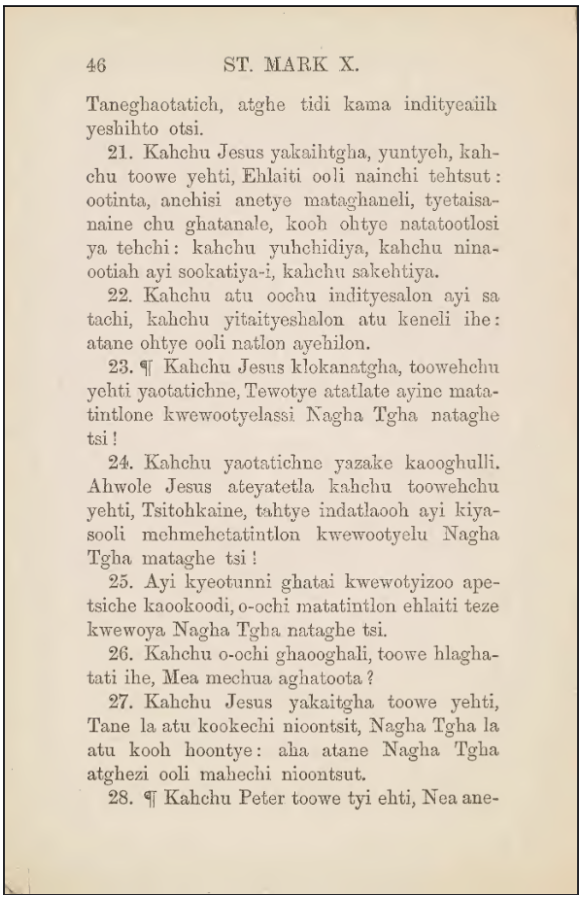
6. Ahwole tahkine ayiti tyechii adesklesne, kahchu koodzye otsi indighateshalon,

7. Yeoogha ti teze atu oochu ateotye? mea metselihi ka oontoolelassi kootahtye Nagha Tgha zon?

8. Kwete Jesus ataotio yaghatei intidesakehe, toowela ooli yehtilon, Yehoo oohtye naghadzye ihe intidaa aoontye?

We attempted to use an optical character recognition (OCR) reader on the pdf document to pull the text from it. What it did extract from the document was messy data with missing characters and also extra characters being picked up from the poor quality of the photos. An example of a messy page and its incorrect results from the OCR reader can be seen in figures 2 and 3 respectively. One of the mistakes can be found in the last word of the second line in figure 2 and 3, can you find any more on this page?

FIGURE 2



46 ST. MARK X.

Taneghaotatich, atghe tidi kama indityeaiih yeshihto otsi.

21. Kahchu Jesus yakaihtgha, yuntyeh, kahchu toowe yehti, Ehlaiti ooli nainchi tehtsut: ootinta, anchisi anetye mataghaneli, tyetaisanaine chu ghatanale, kooh ohtye natatootlosi ya tehchi: kahchu yuhchidiya, kahchu ninaootiah ayi sookatiya-i, kahchu sakehtiya.

22. Kahchu atu oochu indityesalon ayi satachi, kahchu yitaityeshalon atu keneli ihe: atane ohtye ooli natlon ayehilon.

23. ¶ Kahchu Jesus klokanatgha, toowehchu yehti yaotatichne, Tewotye atatlate ayinc matatintlone kwewootyelassi Nagha Tgha nataghe tsi!

24. Kahchu yaotatichne yazake kaooghulli. Ahwole Jesus ateyatetla kahchu toowehchu yehti, Tsitohkaine, tahtye indatlaooh ayi kiyasooli mehmehctatintlon kwewootyelu Nagha Tgha mataghe tsi!

25. Ayi kyeotunni ghatai kwewotyizoo apetsiche kaookoodi, o-ochi matatintlon ehlaiti teze kwewoya Nagha Tgha nataghe tsi.

26. Kahchu o-ochi ghaooghali, toowe hlaghatati ihe, Mea mechua aghatoota?

27. Kahchu Jesus yakaitgha toowe yehti, Tane la atu kookechi nioontsit, Nagha Tgha la atu kooh hoontye: aha atane Nagha Tgha atghezi ooli mahechi nioontsut.

28. ¶ Kahchu Peter toowe tyi ehti, Nea ane-

FIGURE 3

```
ST. MARK X.
Taneghaotatich, atghe tidi kama inclityeaiili
yeshihto otsi.
21. Kahchu Jesus yalæihtgha., yuntych, kah-
chu toowe yehti, Ehlaiti ooli nainchi tehtsut :
ootinto, anehisi anet.ye mat,aghaueli, tyetaisa-
naine chu ghatanalo, kooh ohtyo notatootlosi
ya tehchi : knhchu yuhchidiya, kahchtl nina-
ootiah ayi sookatiya-i, kahchu sakehtiya.
22. Kahchu atil oochu indityesalon n,yi sa
tachi, kahchu yitaityeshalon atu kencli ihe:
atanc ohtye ooli na.tlon ayehilon.
23. Rallchu Jesus kloknnt,tgha., toowehchll
yohti yaotntichne, Tewotye atatlate ayinc mata-
tintlone kvevootyelnsrsi. Na.crha Tgho nataghe
24. Kahchu yaot?btichno yaza,ke kaooghulli.
Ahwole Jesus ateyatetla kn.hclitl toowehchu
yehti, Tsitohkaine, ta,htye indatlaooh ayi kiya-
sooli mehmehctatintlon Naoha
Tgha mataghe tsi
25. Ayi kyeotunni ghat.ai kwewotyizoo ape-
tsiche kaookoodi, o-ochi matatintlon ehlaiti teze
kwewoya Nagha Tgha, nata,ghe tsi.
26. Kahehu 0-ochi ghaooghali, toowe hlagha,-
tali ihe, Mea mechua aghatoota ?
27. Kahchu Jesus yakaitgha toowe yehti,
Tane 18 atu kookechi uioont,sit, Naoha Tolua la
atu kooh hoontye: aha atane Nagha Tglltb
atghezi ooli mahechi niocmtsut.
28. 91 Kahchu Peter toowe tyi ehti, Nea ane-
```

This would not be useful unless someone manually went through the text to fix the mistakes the OCR reader made. Another method would need to be found, but this gave us the idea to search out more bible scripture for a couple of reasons. The Bible has been translated to many languages and even languages only spoken by very remote people in hopes of bringing those people to the religion. The religious aspect is not what is important here but the fact this would provide parallel translations for many languages that had very little other forms of written text. We also had a very

high confidence that this data would be accurate because it is very likely the Church would not want their message to be misconstrued. The last thing that enticed us was the Bible and its translations are all public, so there is no worry of using someone's private data or data you may have to pay for. This led us to a website, www.bible.com, that contained thousands of translations for the Bible, some of which were considered low-resource languages. Basque and Navajo both had translations for sections of the Bible which we wanted to use; the issue was how do we collect it all.

3.1. DATA COLLECTION AND PROCESSING

There were two ways that I saw we could collect this data: manually copy and pasting the data to a text document which could then be processed or create a web scraper for this website. With my two colleagues being linguistics researchers and myself being a computer scientist I was tasked with collecting the data. I chose to use the web scraper and will go though some of the reasons why and the code in this section, the Data Storage, and Additional Features sections. I wanted to use the web scraper because in the long run it would be easier than manually coping the text and would also allow for some preprocessing to be done alongside data collection. Due to the structure of the website, it allowed for a somewhat easy automated data collection process, not that this cannot be done with other websites, but I'll explain what I mean. The website we used defines which Bible we used through a numeric code, the Biblical book, and the language version used all in the URL of the page. This allows us to go directly to the Bible version and chapter we want without needing to go through intermediary pages. With this initial part figured out this meant there were two next steps. How do we get it to move from to the next chapter of the Bible version wanted and what do we actually scrape from the page?

Taking on the problem of moving to the next chapter I anticipated would be more difficult so I started with this one first. After looking at the Bible versions for the two respective languages I noticed that not every chapter of the Bible was translated for the respective languages. To work around this, I obtained a list of the chapters and the abbreviation used in the URL embedded in the HTML of the website. I could then create an array of the abbreviations to use to ping the website in the web scraper. The outermost layers of the web scraper consist of loops used to ping the website by cycling through the list of abbreviations to complete the URL. If a ping was successful this indicates that the Bible version contained the chapter pinged and the HTML of the page would be grabbed, this process will be talked about later. If a ping was not successful this would mean

the chapter is not there and to move onto the next. At this point in the development process the only other feature added to this layer of the code was being able to choose specific ranges of chapters to ping and possibly grab. We wanted to start collecting data in any format so I moved onto the next problem.

3.2. DATA STORAGE

The next major problem was what did we want off of the page and how should the data be stored. The way the data was obtained was through a python package called BeautifulSoup which would grab the entirety of the HTML for the page pinged. After reading through the HTML, I created a regular expression that would pick out the text of the titles and the verses based off of the HTML tags the text would reside in. After getting to this point, I needed a way to store the data. The website would sometimes provide the title of the chapter on the page and also sub-titles depending on the version of the Bible. Another categorizing feature given was the verse numbers of each chapter. Using all of these categorizing features in conjunction with one another provides a simple way to catalog the data and also this would provide enough break down of the text so that it would give an ample amount of data points but also each data point would have some depth to it. The storing of the data was done through text files which consisted of two columns. The first columns were numeric reference consisting of the book version, chapter number, verse number, and an indication if it was a title or the verse itself, these values were separated by a colon. The second column was the cleaned text of the verse. Here the text needed to be cleaned of irregular characters, these were things like commas and dashes that were changed to comply with the machine learning tools being used. At this point the web scraper was working in its most primitive form. It was capable of connecting to the version of the Bible needed if you knew the numeric code, could then collect all the information for that specific bible, and store the information into a more user-friendly version.

3.3. ADDITIONAL FEATURES OF DATA COLLECTION AND PROCESSING

After refining the code somewhat and talking with the other researchers we wanted to add more features to the web scraper. The two major tasks were to make the web scraper more universal so that it could be useable on any of the Bible's available and to also have the preprocessing work be done alongside the initial storing of the data. At this point I opted to give the file command line options to make the web scraping process more user friendly. I added features to scrape all

available bibles, create specific lists of bibles to grab, range of bibles to grab, an option to tokenize words based off English grammar, an option to tokenize based of a standard rule set, and a syllabifier. A more hidden feature available if you want to write some of your own code for a language was to create your own tokenizer. During the preprocessing of data within the web scraper another file gets called to tokenize the text. Within this file tokenizers can be added for specific languages and if one is not present the standard rule set is used unless instructed otherwise. We experimented with a few different types of tokenizers, all of which can still be seen in the original code. At this stage the code was much closer to its final form, from here only bugs and small formatting changes in file structure used to save the text were changed in the code. From here there was nothing left to do but collect our data and begin building our models.

4. CONCLUSION

Now that we were able to collect and preprocess our data, we could actually run the experiments that Mans and Ling had originally set forth. Is it possible to create neural machine translation model from a low-resource language and can any additional techniques be applied to help aid this? The results of our research can be found in our paper (Liu et al. 2021) and the code itself can be seen on our GitHub (Liu et al. 2021).

REFERENCES

Conneau, Alexis; Guillaume Lample; Marc'Aurelio Ranzato; Ludovic Denoyer; and Hervé Jégou. Word Translation without Parallel Data. arXiv:1710.04087 [cs.CL] (January 30, 2018). Accessed April, 2021. Online: http://arxiv.org/abs/1710.04087.

Read the Bible. A free Bible on your phone, tablet, and computer. Read the Bible. A free Bible on your phone, tablet, and computer. | The Bible App | Bible.com. (n.d.). https://www.bible.com/.

Liu, L., Ryan, Z., & Hulden, M. (2021). The Usefulness of Bibles in Low-Resource Machine Translation. Proceedings of the Workshop on Computational Methods for Endangered Languages, 1, 44–50. https://doi.org/10.33011/computel.v1i.957

Liu, L., Ryan, Z., & Hulden, M. (2021). The Usefulness of Bibles in Low-Resource Machine Translation. GitHub Repository, https://github.com/LonelyRider-cs/Low_Resource_MT