Global Classroom: Predictive Text for Chukchi

Evan Chapple
Indiana University
Bloomington

Elizabeth Gabel
Indiana University
Bloomington

Emil Nadimanov
Higher School of
Economics Moscow

Tatiana Yudina
Higher School of
Economics Moscow

Abstract

In this paper we discuss a system of text prediction for the indigenous Siberian language Chukchi. Chukchi is low-resource, and the corpus provided for this project was small. Chukchi is polysynthetic and has a variety of distinct dialects, which increases the impact of a smaller corpus. We discuss the effect of small corpora on training, and discuss the necessity of redefining what qualifies as a useful unit of meaning, depending on the language.

The system introduced in this paper is a bi-gram system which uses analysis on the morpheme level rather than the word level. This accounts for the differences in the requirements for English-oriented predictive systems and Chukchi's syntactic structure. A test for accuracy is run on the system, and evaluated for improvement from the original word-level analysis provided in the baseline.

1 Introduction

The purpose of this paper is to describe a predictive text system for Chukchi, an endangered indigenous language spoken in northeastern Siberia. The system is meant to improve on baseline code provided for the project. Our text predictor is comprised of two bi-gram models which are passed the beginning of a "word" and attempt to determine the most likely morpheme or character given frequencies based on the training data.

Chukchi is a morphologically rich language, with words comprised of several different morphemes. A single word can contain meaning which is conveyed in an entire sentence in another language (i.e., English). This means that where word-level analysis might be effective in English, it is not useful in text prediction for Chukchi. We instead analyze frequencies at the morpheme level, changing our "word" to a morpheme and then, if necessary, a character.

Languages such as English are spoken by a large number of people, so there are large data sets to train programs on, with programs frequently training on billions of tokens ((Boudreau et al., 2020)). The smaller size of the Chukchi corpora used to train our system limited the accuracy of the predictions. We used two corpora, one provided to us and a larger one which we developed on our own. Thus, this paper discusses both the benefits of analyzing Chukchi at the morpheme level and the benefits of large corpora for training.

The second section will discuss relevant literature, including features of Chukchi, an explanation of N-grams, and how we define a "word" for the purposes of this paper. The third section will discuss our training data. The fourth section will discuss our system, the fifth section our results, and the sixth section will discuss our potential future work.

2 Related Work

2.1 Chukchi Language

Chukchi is an indigenous language spoken in northeastern Siberia. A member of the Chukotko-Kamchatkan family, it is a polysynthetic language featuring vowel harmony and both syntactic and morphological ergativity (Kantarovich, 2020). A variety of Chukchi dialects are recognized, including the separate dialects of male and female speakers discussed in Dunn, 1999.

Chukchi is considered highly-endangered, with approximately 5000 speakers, with few children learning the language (Andriyanets and Tyers, 2018). It's orthography is based on the Cyrillic alphabet, although Dunn argues that the language is ill-suited to this orthographic system, with redundancies and Russian spelling conventions which do

not apply to Chukchi(Dunn, 1999).

Few grammars have been written for Chukchi, and none of them have been complete. Bogoras's 1922 grammar doesn't account for syntax and Skorik's grammars describe an artificial version of the language which, while easier to describe under Indo-European conventions, does not represent the state of Chukchi as actually spoken. Dunn's 1999 grammar is comprehensive, but only focuses on a single dialect, Telqep, and has limited examples of each feature of Chukchi discussed (Kantarovich, 2020). This lack of detailed grammars and reduced amount of Chukchi resources make Chukchi a low-resource language.

Chukchi has dominant and recessive vowels which are determined based on vowel height, and the vowel agreement in Chukchi is both progressive and regressive (Andriyanets and Tyers, 2018). Regressive vowel agreement makes morpheme prediction difficult, since a system can only predict what is next based on previous characters. Chukchi also has a variety of affixes, with the variation in affixes sometimes depending on who the speaker is (i.e., older or younger, male or female) (Kantarovich, 2020). The variability in declensions, conjugations, and affixes in general added an extra layer of difficulty to this project, as there are more environments which the system has to be trained on to be able to accurately assess which morpheme is most likely.

2.2 N-gram models

The idea behind N-grams is one that is pretty simple. It involves the prediction of the next most probable word/morpheme/character, A_1, given a word/morpheme/character, A_0 (Jurafsky and Martin, 2020). These N-grams are built by training the model on a corpus of data then applying it to another set. The model will be trained in such a way as to create tuples of a word and their probability given another word. In order to find the next most probable word, all we have to do is take our key, A_0, and then choose the tuple with the highest probability associated with it.

2.3 Defining a "word"

As seen in the previous section, these methods can be applied to words, morphemes, and characters. Ünstün et al., 2018 examines the success of n-gram models in different languages. The most applicable finding of theirs is that analysis done on the morpheme level rather than the word level, is far more effective in languages that are morphologically rich. These findings directly influenced our methodology. There are multiple applications of n-grams which will be discussed throughout this paper and in the future work section.

3 Resources

The data used to train and test our system was provided to us for the project. The data was supplied in a two column format, the first of which provided the orthography used to write normally, and the second which showed words split at their morpheme boundaries with a morpheme boundary marker ">". The training data supplied consisted of 30,000 lines with an additional practice test set of 1,000 lines. An example of our training data is available in Figure 1.

нивқин пэтӆе нэмэ	н>ив>қин пэтӆ>е нэмэ
рэпиӈэръугъэ мэӆқырым	рэпиӈэ>ръу>гъ>э мэӆқырым
татӆыгнэн савинына игыр	тат>ӆыг>нэн сави>ны>н>а игыр
вэтықун мытрэриӈэгъэ	вэты>қун мыт>рэ>риӈэ>гъ>э
пэтӆе пэтӆ>е	

Figure 1: Excerpt of the formatted data used in model training.

The provided corpus (retrieved from this website) is fairly small, limiting how effective it is in training our system. In recognition of the potential shortcomings of the provided corpus, we also used an extended corpus (referred to as "big corpus" or "BC" throughout) compiled by one of our members. The BC is comprised of Chukchi language salvaged from a variety of resources ranging from Chukchi fairy tales to newspapers. These sources are usually not formatted at morpheme boundaries, so were not of much use on that front. In the results section we provide the results for our system when trained on both the provided corpus (PC) and the BC.

4 Language Model

The implementation used in our project is made up of two, relatively simple parts. The first of which is a character bi-gram model and the second is a morpheme bi-gram model. The first is only implemented when the input supplied does not lead to the successful finding of a morpheme.

The morpheme bi-gram model that we implemented first runs our training data, and collects all bi-grams of morphemes present in the data. It then assigns the pairs a probability——occurrences of morpheme A_0 followed by morpheme A_1 divided by occurrences of morpheme A_0. The formula is supplied below for the sake of clarity. Then, when running the prediction part of the program, once there are enough characters supplied to find a mor-

pheme, it finds the bi-gram pair that has the highest probability and guesses it. Guessing checks whether the prediction is correct. If our prediction is not correct, the program can not guess again, so we continue and try to predict subsequent characters using the process described below.

$$P(a, x) = \frac{occurrences\ of\ x\ after\ a}{total\ occurrences\ of\ a} \quad (1)$$

The character bi-gram model serves as a fallback for the program and does not result in as many substantial predictions. Our bi-gram model collects bigrams of characters and, given a character, C_0, will find the character, C_1, with the highest probability of following C_0. These parts of the program are accessed during different steps of running. The first is accessed when there are enough characters to find a morpheme in our morpheme bank and the second when an incorrect morpheme is guessed and the rest of the word does not permit the use of the morpheme guesser.

Below, we will investigate how our system deals with a character-string input and see how the two layers of bi-gram predictions pass and interact. The full string, along with its correct morpheme boundaries is, "пэтӆе пэтӆ>є".

INPUT	Morph.	Guess	Char.	Guess	Success
п	✓	-ъым	N/A	N/A	False
пэ	N/A	N/A	✓	-т	True
пэт	✓	-ӆе	N/A	N/A	False
пэтӆ	✓	-е	N/A	N/A	True

Table A: Demonstration of a simple Chukchi word and the predictions provided by the program

For the sake of clarity, "Morph." refers to the morpheme bi-gram prediction and "Char." refers to the character bi-gram prediction. When provided with the first character for input, the morpheme bigram finds a match in its list and attempts to guess the most common continuation. Since, this prediction is wrong, the user now enters another character. The program does not find this new two character-long string so the morpheme prediction is passed, and the program then relies on the character bigram prediction. This prediction is correct, therefore a hit is counted. The program is now provided with a three character-long string. This string does result in a morpheme level prediction, however, it is not right. Another character is then added to the input string. This string then allows a morpheme level prediction which returns the correct, following character.

In previous iterations of the program we tested the efficacy of higher grams, including tri-grams. We found that this method performed worse than the bi-gram model. It is possible that our data was not large enough to build accurate prediction models, and thus very rare combinations were returned more often than they should have been.

5 Results

Prior to receiving the data on which our predictor's accuracy would be graded, we trained our system on both the PC and the BC, testing its accuracy on provided development data. Table A shows these results.

	Base PC	bi-gram PC	bi-gram BC
Char	37897	37897	37897
Tokens	8788	8788	8788
Clicks	37754	34849	31598
Clicks/Token	4.296	3.966	3.596
Clicks/Char	.99623	.91957	.83379

Table B: Results from development data

We then received the test data on which our system would be evaluated. We used PC and BC for training again.

	Base PC	bi-gram PC	bi-gram BC
Char	37927	37927	37927
Tokens	8374	8374	8374
Clicks	37768	36602	36291
Clicks/Token	4.510	4.371	4.334
Clicks/Char	.99581	.96506	.95686

Table C: Results from test data

The difference between the base PC and bi-gram PC is clear, with the bi-gram model trained on the PC performing better that the base trained on the PC. This is true for both Clicks/Token and Clicks/Character, showing that the system is more accurate overall. This is true for both the developmental and the test data, showing that regardless of the data-set, the bi-gram model outperforms the baseline. This indicates that adding a level of subword information and analysis in the form of morphological analysis increases the accuracy of text prediction for Chukchi.

The difference between the BC and PC is also clear, with the model trained on the BC outperforming the PC. This is despite the PC's words be-

ing demarcated at the morpheme boundaries. Even though cleaner data is generally accepted as more accurate, the fact that the BC increased the accuracy of our model's predictions indicates that larger corpora yield more accurate results.

5.1 Sources of Error

The large difference between the development data (DD) and the test data (TD) raises questions about what the differences between the two corpora might be. One potential explanation is that the two corpora might be written in different dialects, with one easier to tokenize than the other. It's also possible that the corpus is of relatively poor quality, lacking uniformity which contributes to some morphemes which are the same being classified as different.

We also had our DD for the entire design process. It is very likely that we subtly oriented our system to the data which we were given, making small changes which improved the results without improving the actual ability of the system to make predictions.

6 Future Work

6.1 Word Embedding

There are certain extensions to our model that could lead it to be more effective. One way to do so is word embedding. Word embedding, in short, is the process of translating words into vectors. There are basic ways to do this encoding such as one-shot, but since this method does not preserve or incorporate context into the model, it is not very useful in next-word/morpheme prediction.

There are other systems that have been developed with preserving this context in mind. Some of the most well known of these are Word2Vec and Fasttext, an extension of Word2Vec (Bojanowski et al., 2017). These models involve storing the individual words as vectors based on the context that surround them, defined by a specific window size. There are two ways to train a Word2Vec model which are either Skip-Gram or Continuous Bag of Words (CBOW). These models only differ on what is supplied as input and what is returned as output, either the context surrounding a word or the word itself. The CBOW model seems most relevant to word prediction as we want to supply a certain context, that which has been fed into so far, and output the most likely word/morpheme to fit that context. FastText defines a word in an interesting way, maybe more akin to a morpheme but not quite. FastText splits words and even sentences based on an n-gram size and ignores word boundaries. Then, it defines the vector of a particular word as the summation of all the vectors that make up its parts. One downside of this particular approach is that certain semantic meanings are lost if the surface forms of the words are disparate, however, this does not pose a problem for next word/morpheme prediction. These types of systems could be applied to the exact problem this paper attempts to address, and possibly could result in higher success rates.

We can also consider the benefits of supervised machine learning as applied to embedding. An example of this is predictive text embedding, or PTE, as discussed by Tang et al.,2015. This method of text embedding uses partial tagging and then embeds the text into a heterogeneous text network which shares vectors across three bipartite networks to increase the accuracy of analysis, drawing from both the labeled and unlabeled pool to train. This method makes text embedding more competitive in certain machine learning tasks when compared with neural networks.

When training these specific programs on Chukchi, the thought is that these models would be best run when the input defines a single Chukchi word as a sentence and all the morphemes inside it, words. This approach was seen in (Ünstün et al., 2018), and it was shown to have positive results for agglutinative languages.

6.2 Tries

There is also a different way to search through our list of morphemes than checking every string linearly. This would be by using a trie (D.E., 1973). Tries are data structures used to store and return strings. To explain how a trie works, we first need to understand how it is structured and to do this, we will think about English. A trie has the same number of levels as the maximum length of a string stored there within. The first node, also the topmost, has a null value and points to up to 26 other nodes at level 2, 26 because English strings only have 26 possible characters in the first position. All of the nodes in level 2 should store a single character. Next, all of those 26 nodes can point to 26 more nodes below it, in level 3. In level 3, all nodes should store two characters and so on, and so forth. When one is then searching for a value within this tree, we can add each additional character and move down the tree until we find a matching bucket. This type of structure should lower the maximum possible run

time for the program.

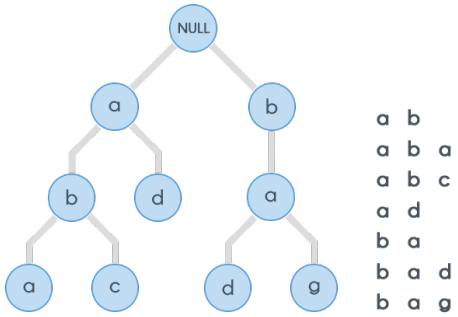

Figure 2: Example of what is stored at various levels in a string trie tree (Tri)

7 Conclusion

In this paper we described a predictive text system for Chukchi which predicts text based off a two-pronged analysis at the morpheme level. The system first trains on data to determine which morpheme and character combinations are most frequent, and then uses the frequencies determined from the training to create bi-grams. The system uses this data to attempt to guess the most likely morpheme to follow a provided morpheme in the text. When the system cannot guess the morpheme, it then moves to character prediction, allowing the system to eventually guess another morpheme. We discussed how efficient the system is, the value of large corpora, and some of the difficulty training a system on a low-resource language. All of our code is available on our github[1]. Our results indicate that morpheme-level analysis is more effective for Chukchi, and we mention ways to improve and continue this research.

References

Trie (Keyword Tree). Hacker Earth. https://www.hackerearth.com/practice/data-structures/advanced-data-structures/trie-keyword-tree/tutorial/.

Vasilisa Andriyanets and Francis M. Tyers. 2018. A prototype finite-state morphological analyser for chukchi. *Proceedings of the Workshop on Computational Modeling of Polysynthetic Languages*, pages 31–40.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information.

Jeremie Boudreau, Akankshya Patra, Ashima Suvarna, and Paul Cook. 2020. Evaluating the impact of sub-word information and cross-lingual word embeddings on mi'kmaq language modelling. *Proceedings of the 12th Language Resources and Evaluation Conference*. https://www.aclweb.org/anthology/2020.lrec-1.333.

Knuth D.E. 1973. *The Art of Computer Programming (Volume 3)*. Addison-Wesley Publishing Company.

Michael J. Dunn. 1999. A grammar of chukchi. *PhD Thesis, Australian National University*.

Daniel Jurafsky and James H. Martin. 2020. N-gram language models. *Speech and Language Processing*.

Jessica Kantarovich. 2020. Argument structure in language shift: Morphosyntactic variation and grammatical resilience in modern chukchi. *PhD Thesis, University of Chicago*.

William Lane and Steve Bird. 2020. Bootstrapping techniques for polysynthetic morphological analysis. https://drive.google.com/file/d/1c0zwikNd6e9LfJrQzPbBPXJtTN_3TCo0/view?usp=sharing.

J. Tang, M. Qu, and Q. Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. *KDD '15: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 1165–1174.

Ahmet Ünstün, Murathan Kurfah, and Burcu Can. 2018. Characters or morphemes: How to represent words? *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 144–153.

[1]The code and parameters of the project are available at our github