ÚFAL Submission for SIGTYP Supervised Cognate Detection Task

Tomasz Limisiewicz
Faculty of Mathematics and Physics, Charles University in Prague
limisiewcz@ufal.mff.uni.cz

Abstract

In this work, I present ÚFAL submission for the supervised task of detecting cognates and derivatives. Cognates are word pairs in different languages sharing the origin in earlier attested forms in ancestral language, while derivatives come directly from another language. For the task, I developed gradient boosted tree classifier trained on linguistic and statistical features. The solution came first from two delivered systems with an 87% F1 score on the test split. This write-up gives an insight into the system and shows the importance of using linguistic features and character-level statistics for the task.

1 Introduction

The described system is a supervised model trained for three-way classifications aimed to distinguish cognate and cross-lingual derivatives or no relationship for pairs of words in different languages. Cognates are pairs with similar meanings and come from the same root in an ancestral language. For instance, the German "vater" is cognate with the English "father" coming from the same Proto-Indo-European root. In contrast, multilingual derivatives are words borrowed from another language potentially with some modification, e.g., the word "restaurant" in English comes from a French word with the same spelling (Crystal, 2008).

The solution used only the data provided by the organizers, i.e., 232,482 bilingual pairs in 34 European languages. The data came with the relationship labels (cognate, derivative, or no relation) scraped from Wiktionary.[1] In the examples containing derivative pairs, the order of words did not indicate the source and recipient language.

The proposed system was evaluated on the test data with 876 bilingual word pairs with hidden target labels. The evaluation metric was a macro-averaged F1 score. For development purposes, I sampled 10% of the provided training data to create a validation set not used in the model fitting.

My solution is based on gradient boosted tree classifier trained on the set of language features comprising multilingual language model embeddings, language and language group id, character-level Levenshtein distance, and a binary variable marking capitalized words.

The system obtained the F1 score of **87%** on the test set and came first out of two submitted to the shared task. The source code for the submission is publicly available at GitHub: https://github.com/tomlimi/cognate_detection.

The system description is organized in the following way: in Section 2, I describe the classification model and the hyperparameter search method; in Section 3, I introduce the features selected as input for the classifier; lastly, in Section 4, I present the results of the method together with the accumulation study and the analysis of feature importance.

2 Classification

For classification tasks, I used a gradient-boosted tree implemented in the XGBoost library (Chen and Guestrin, 2016).[2] The boosting tree is the method that enables the predictions of a large set of decision trees obtained with a gradient search. This section describes the hyperparameters used for the classifier and the method used to select them.

I chose XGBoost because it performs well for data containing real and discrete variables, and the set of input variables can be easily extended. Moreover, XGBoost can be interpreted through feature importance analysis.

2.1 Class Weighting

The cognate data were significantly skewed toward no relation class (78.0%), followed by derived pairs (16.9%) and cognates (5.1%). The task organizers

[1] https://www.wiktionary.org/

[2] https://xgboost.readthedocs.io/

Proceedings of the 5th Workshop on Research in Computational Linguistic Typology and Multilingual NLP (SIGTYP 2023), pages 132–136

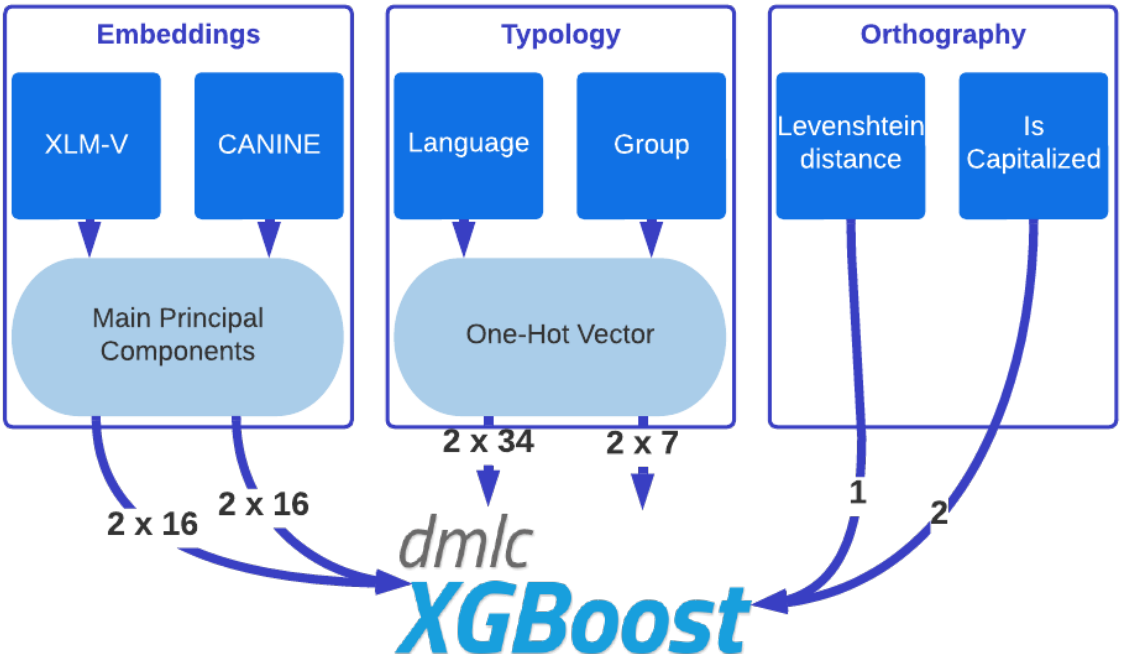

Figure 1: The visualization of features selected as an input to XGBoost classifier. For each word from the test cases, I obtained embeddings from two models, XLM-V and CANINE-C. The embeddings were compressed to 16 dimensions for each model by selecting the first principal components. The language and language group labels were fed to the classifier as one-hot vectors. Levenshtein distance between words in each pair was inputted as a real-valued variable. The last component of the classifier input were binary variables denoting if each word of the pair is capitalized.

notified contestants that the training data contains a significant share of false positives, i.e., unmarked cognates pairs. For those reasons, I weighted test examples in order to counter the imbalance, assigning higher weights to examples containing cognate and derived forms.

2.2 XGBoost Hyperparameters

The boosting algorithm was trained to maximize the area under the classification curve with gradient descent performed for 100 steps. In the parameter search, I considered the following ones:

- **eta** shrinks the weights of features.
- **gamma** minimum loss reduction needed to make a partition of the node
- **maximum depth** maximum depth of the tree.
- **minimum child weight** minimum sum of the instance weights in a leaf.
- **maximum delta step** the cap of the output in the leaf helps to counter data imbalance
- **subsample** sampling training instances for each boosting iteration.
- **column sample** family of arguments: sampling columns (features) before adding a new tree, level, or node.
- **lambda** L2 regularization on the model's weights.
- **alpha** L1 regularization on the model's weights.

2.3 Bayesian Paremeter Search

The hyperparameters are searched by Bayesian optimization (Bergstra et al., 2013) based on the Hyperopt library.[3] In this algorithm, the hyperparameter space is searched by sampling the configuration with a high probability of increasing the objective function. The search is performed iteratively, updating hyperparameter distributions after each epoch. I ran a Bayesian search for 50 epochs (in each epoch, the XGBoost was run for 100 steps).

[3] https://hyperopt.github.io/hyperopt

Parameter	Search Range	Selected
eta	0.01 - 0.3	0.275
gamma	0 - 5.0	0.642
maximum depth	3 - 20	12
minimum child weight	1 - 6	4
subsample	0.6 - 1.0	0.723
column sample (tree)	0.6 - 1.0	0.919
column sample (node)	0.6 - 1.0	0.749
column sample (level)	0.6 - 1.0	0.998
lambda	0 - 5.0	1.507
alpha	0 - 5.0	1.138

Table 1: Hyperparameter search spaces: uniform distributions in the given ranges. The value was selected with Bayesian optimization. Distributions of maximum depth and minimum child weight are discrete.

Table 1 shows the search spaces and selected parameters.

3 Feature Selection

I used an ensemble of word embeddings, typological and orthographical information as input to the XGBoost classifier. This Section describes how those features were selected and pre-processed. The visualization of all the picked features is presented in Figure 1.

3.1 Language Model Embeddings

I computed the embedding representation of the words in each test pair. I took the final layer representation of two recent multilingual Transformer-based models available through the HuggigFace interface (Wolf et al., 2020):[4] **XLM-V** (Liang et al., 2023) and **CANINE** (Clark et al., 2022). The former model tokenizes the input with a large (1 million entries) subword vocabulary. The latter splits the input into character sequences and applies a convolution layer before the proper Transformer. I used these two models aiming to merge character and subword signals.

The resulting word embeddings have high dimensionality, i.e., 1024 for each model. I decided to decrease the dimensionality of the embeddings in order to balance out the composition of the classifier input vector. For that purpose, I applied SVD decomposition on the embeddings obtained for the training set and sorted the principal components in the order of the variance explained. Subsequently, I picked the first 16 principal components for each model and used the projection matrix to obtain the representation for the development and test sets.

[4] https://huggingface.co/

	Features	Train Acc	Train F1	Validation Acc	Validation F1
1	Language ID	75.9	64.2	76.1	64.2
2	(1) + Group ID	76.6	64.7	76.7	64.6
3	(2) + Capitalized	78.4	66.3	78.6	66.4
4	(3) + Levenshtein	83.1	70.6	83.0	69.8
5	(3) + Embeddings	97.2	94.2	92.6	80.3
6	(4) + Embeddings No weighting	**98.4**	95.8	**93.8**	79.6
7	**(4) + Embeddings**	97.8	**95.3**	93.7	**82.7**

Table 2: The feature accumulation analysis results from the XGBoost classifier. Each row presents the results for the model trained on a different set of features.

3.2 Typology

I encoded language information as two class variables: the first is **language identity** (34 classes), and the second is **language group identity** (7 classes: Romance, Slavic, Germanic, Celtic, Hellenic, Baltic).

Both variables were encoded in a one-hot vector, with 34 dimensions for language and 7 dimensions for language family.

3.3 Orthography

I used **Levensthein distance** (Levenshtein, 1966) on character level as the measure of similarity between words. The second feature based on orthographical forms was the binary variable denoting for each word whether it **is capitalized**. I added this feature because I have observed that proper names are often borrowed in other languages. Therefore, the capitalized word's appearance increased the derivative class's probability.

Admittedly, the adequate way to utilize Levenshtein distance would be to compute it on the phoneme level. However, the text-to-phonemes models were not publicly available for many low-resource languages included in the shared task.

4 Results

I trained the classifier on top of input vectors constructed from the features described in Section 3 and using the hyperparameters picked by Bayesian search described in Section 2.3. I split the training

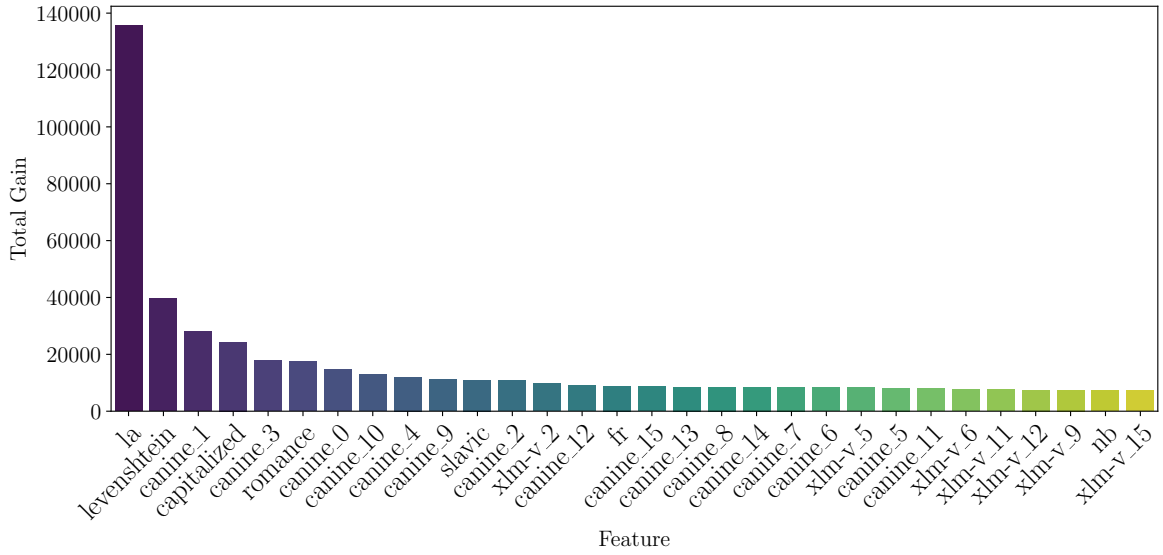

Figure 2: The sum of objective gains for when a given feature was used for a tree split. Features *la*, *fr*, *nb* are ones of 34 language ID features indicating if one of the languages in the pair is Latin, French, or Bokmål (Norwegian). *canine_x* and *xlm-v_x* denote the main principal component of language models CANINE and XLM-V, where *x* indicates the rank of the component according to the proportion of explained variance. The figure presents only 20 features with the highest total gain.

set provided by the organizers into train and validation splits containing 90% and 10% randomly selected data examples.

The submitted solution obtained **95.3%** macro F1 score on the train set and **82.7%** on the validation set. On the held-out test split, the system achieved **87%** F1 score, as reported by the organizers. This section presents the results of the accumulation study and feature importance analysis.

4.1 Accumulation Analysis

Table 2 shows the accuracies obtained by the classifier trained on subsets of features. Interestingly, the classifier trained just on language labels achieves a relatively high F1 (64.2% on the validation set). The highest gain is observed after adding word embeddings (+12.9% validation F1 increase in (8)); Levenshtein distance also visibly improves results (+3.87% in (5)). The model without class weighting (7) achieves better class accuracies and a lower F1 score due to class imbalance.

In summary: there is a visible impact of including language model embeddings and Levenshtein distance as classification features.

4.2 Feature Importance

Figure 2 presents the feature importance computed as the total gain each feature brought in the splits.

The most important feature is a binary variable indicating if one of the languages in a pair is Latin. The importance of this feature can be explained by the fact that Latin is the source language of many borrowings throughout European languages. The second feature is Levesnshtein distance, followed by one of the CANINE principal components (*canine_1*) and binary variable marking capitalized words. These three feature depends on the character composition of the analyzed words, highlighting the importance of orthographical information for cognate detection. Furthermore, character-based CANINE embeddings tend to be more influential to the predictions than subword-based ones (XLM-V).

5 Conclusions

The developed supervised system achieves competitive results in cognate detection (87% on the test set). The model was trained on a diverse set of linguistic and statistical signals. The accumulation and importance analysis showed the importance of nuance aspects of the dataset, such as Latin or capitalization, as an indication of derivative relation. The analysis also showed the high importance of using character-based representation for the task in the form of CANINE embedding and character-level Levenshtein distance.

Limitations

I acknowledge that the solution is limited in its scope. For instance, I did not use phonetical representation, which is more suitable for comparing the potential cognates across languages. Also, the solution could benefit from more complex historical linguistic analysis, e.g., obtained with Pyling package (List and Forkel, 2021).[5] However, the proposed classification method can be easily extended to incorporate additional features.

The flawed annotation of the training set causes another limitation of the method. According to information from the organizers, the dataset contained a significant number of false negatives, i.e., missing cognate relations.

Acknowledgments

I thank Abishek Stephen for his theoretical insight and valuable suggestions for using linguistic features in the classification model. I also thank Martin Popel, Ondřej Plátek, and Ondřej Dušek for their helpful comments on the previous draft of this system description. My work has been supported by grant 338521 of the Charles University Grant Agency.

References

James Bergstra, Daniel Yamins, and David Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 115–123, Atlanta, Georgia, USA. PMLR.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an Efficient Tokenization-Free Encoder for Language Representation. *Trans. Assoc. Comput. Linguistics*, 10:73–91.

David Crystal. 2008. *A Dictionary of Linguistics and Phonetics*, 6th ed. edition. Blackwell Pub Malden, MA ; Oxford.

V. I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Cybern Control Theory*, 10:707–710.

Davis Liang, Hila Gonen, Yuning Mao, Rui Hou, Naman Goyal, Marjan Ghazvininejad, Luke Zettlemoyer, and Madian Khabsa. 2023. XLM-V: Overcoming the Vocabulary Bottleneck in Multilingual Masked Language Models. *CoRR*, abs/2301.10472.

Johann-Mattis List and Robert Forkel. 2021. LingPy. A Python Library for Historical Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics.

[5] https://lingpy.readthedocs.io