Morphology-Aware Meta-Embeddings for Tamil

Arjun Krishnan*
Princeton University
ak36@princeton.edu

Seyoon Ragavan*
Princeton University
sragavan@princeton.edu

Abstract

In this work, we explore generating morphologically enhanced word embeddings for Tamil, a highly agglutinative South Indian language with rich morphology that remains low-resource with regards to NLP tasks. We present here the first-ever word analogy dataset for Tamil, consisting of 4499 hand-curated word tetrads across 10 semantic and 13 morphological relation types. Using a rules-based morphological segmenter and meta-embedding techniques, we train meta-embeddings that outperform existing baselines by 16% on our analogy task and appear to mitigate a previously observed trade-off between semantic and morphological accuracy.

1 Introduction

Continuous-space word embedding methods such as word2vec (Mikolov et al., 2013) have proven to be very useful for a wide range of NLP tasks. However, it has been observed that representations that treat each word holistically face inherent limitations when working with morphologically rich languages, and methods have accordingly been designed to incorporate subword information (Cotterell and Schütze, 2015; Luong et al., 2013). Among these, the fastText embeddings remain one of the best-known (Bojanowski et al., 2017; Grave et al., 2018), using character n-grams to approximate word-internal structural features.

In this work, we focus on producing morphology-aware embeddings for Tamil, a Dravidian language with over 68 million speakers across India, Sri Lanka, Malaysia, and Singapore (Wikipedia, 2020). Tamil remains a low-resource language for NLP tasks despite its large speaker base, and traditional methods of evaluating word embeddings, for instance the word analogy task (Mikolov et al., 2013), are almost entirely lacking. Thus, to facilitate our work, we present here a novel, human-curated analogy dataset consisting of 4499 analogy tetrads.

With regard to morphology, Tamil is highly agglutinative, encoding grammatical features such as gender, number, and case in single words comprising large sequences of compounded morphemes. Approaches such as character n-grams that generically incorporate subword information may be too coarse when working with Tamil, due to short morpheme lengths paired with high similarity between morphemes and sandhi across morpheme boundaries. The high frequency of 'false morphemes' or character sequences resembling morphemes in non-productive situations compounds this further. In our work, we attempt to tailor embeddings to Tamil morphology with the incorporation of a rules-based morphological segmenter.

We present three primary contributions:

(1) We present the first-ever human-generated analogy dataset for Tamil, capturing both semantic and morphological analogies.

(2) We construct a set of novel word embeddings for Tamil that incorporates morphological segmentation and outperforms existing baselines trained on the same corpus.

(3) Finally, we show that meta-embedding methods used in conjunction with linear dimension reduction can mitigate previously observed trade-offs between capturing semantic and morphological/syntactic information in embeddings (Avraham and Goldberg, 2017; Qiu et al., 2014).

Our dataset, embeddings, and experiments are all publicly available with documentation at our GitHub repository.

2 Related Work

Explicit morphology in word representations. Explicit incorporation of morphology is limited

*Equal contribution.

Proceedings of NAACL-HLT 2021: Student Research Workshop, pages 94–111
June 6–11, 2021.

by the need for accurate morphological annotation – Luong et al. (2013) used a recursive neural network to combine learned representations of individual morphemes, using the toolkit Morfessor (Creutz and Lagus, 2007) for unsupervised morphological segmentation. Cotterell and Schütze (2015) utilized a hand-annotated, morphologically-labelled corpus to train embeddings to predict morphological tags and thereby encode morphology.

Morphology for low-resource languages. There has been some recent work focusing on morphological incorporation for low-resource agglutinative languages such as Turkish and Uyghur (Pan et al., 2020). Kumar et al. (2017) focused entirely on Dravidian languages, creating a corpus partially annotated for morphological segmentation and POS tagging.

Meta-embeddings. It has been observed that various methods of combining word embeddings into *meta-embeddings* can combine the strengths of individual embeddings to improve performance. Such methods include concatenation (Yin and Schütze, 2016), averaging or summing (Coates and Bollegala, 2018), and constructing a vector of complex numbers (Wittek et al., 2013).

Dimension reduction for word representations. Yin and Schütze (2016) also observed that PCA could reduce the dimension of meta-embeddings without significantly hurting performance. In a similar vein, Mu and Viswanath (2018) found that removing the top few principal components improved performance. Raunak et al. (2019) found that composing these two methods improved dimension reduction, often producing embeddings that even outperformed the original embeddings.

Tamil word embeddings. Tamil word embeddings remain a relatively under-explored space in the literature. The well-known fastText embeddings contain Tamil embeddings in both iterations (Bojanowski et al., 2017; Grave et al., 2018), and represent the state-of-the-art. Kumar et al. (2020) produced a range of embeddings using conventional methods on corpora they produced for 14 Indian languages including Tamil.

Word analogy datasets. The state-of-the-art word analogy dataset in English remains the Google analogy test set developed by Mikolov et al. (2013). Similar datasets have been produced for Spanish (Cardellino, 2016), Russian and Arabic (Abdou et al., 2018), and Chinese (Jin and Wu, 2012), among others. Hindi is the only South Asian language with a high-quality word analogy dataset to date (Grave et al., 2018), which incorporates particular forms of culturally-linked analogies such as kinship terms. In our work, we attempt to similarly capture language-specific analogies for Tamil.

3 Model

3.1 Atomization

Given that we are considering methods that decompose a word into either character n-grams or morphemes, we consider both of these as special cases of decomposing a word into constituent 'atoms', a process we call 'atomization'. An atomizer takes in a word w and outputs a sequence $S(w)$ of atoms. We detail the two main atomizers used in generating our models here:

(1) The first, henceforth called *character 5-grams*, follows the original fastText papers (Bojanowski et al., 2017; Grave et al., 2018), in which a word's atoms are its character 5-grams, as well as the entire word itself.

(2) In our second method, which we designate *morphemes + stem (1-3)-grams*, we modify a pre-existing Tamil stemmer [1] into a rules-based morphological segmenter using `sbl2py` [2]. The segmenter's role is to map each word to its stem and its sequence of morphemes. A word's atoms are then its stem, constituent morphemes, and character (1-3)-grams of the stem. We will closely examine the segmenter's behavior in section 3.4.

3.2 Training

Our setup slightly extends that of Bojanowski et al. (2017), allowing atoms produced by any atomization method to fulfil the role played by character n-grams in the original paper. The model's trainable parameters are the embeddings z_a for the individual atoms and the output vectors v'_w. Following Bojanowski et al. (2017), we sum the atom vectors to obtain the input vector for the word:

$$v_w = \sum_{a \in S(w)} z_a$$

[1] https://github.com/rdamodharan/tamil-stemmer
[2] https://github.com/torfsen/sbl2py

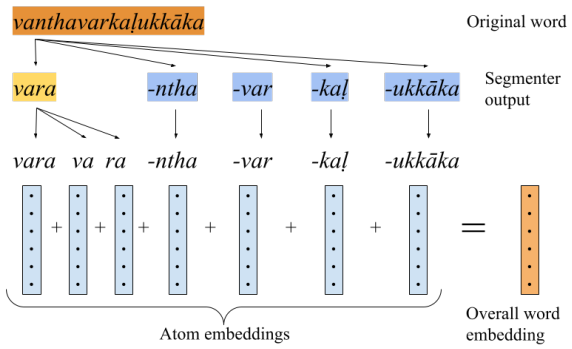

Figure 1: A visualization of a single word's embedding with morphemes + stem (1-3)-grams atomization. The segmenter breaks the word down into morphemes, which together with (1-3)-grams of the stem are our final atoms. The sum of the atom embeddings (which are updated throughout training) is the overall embedding.

The relationship between atom embeddings and the overall word embeddings in training are visualized in Figure 1. Each word has its own output vector that does not depend on the atoms. Using a large text corpus (in this case Wikipedia), we train these embeddings with the skip-gram objective and negative sampling applied to the input and output vectors v_w, v'_w as in (Mikolov et al., 2013).

3.3 Constructing meta-embeddings

Our key primitive for constructing meta-embeddings is a merging operation (Algorithm 1) that takes two separate sets of d-dimensional word embeddings as input and outputs another set of d-dimensional embeddings. It does this by concatenating the two sets of embeddings, then applying PCA to obtain the desired dimensionality. This procedure is visualized in Figure 2.

Algorithm 1: Merge(X_1, X_2)

Data: Embedding matrices
$X_1, X_2 \in \mathbb{R}^{n \times d}$
Result: A new meta-embedding $X \in \mathbb{R}^{n \times d}$
// Rescale to norm 1
$X_1 = \text{NormaliseToUnitNorm}(X_1)$;
$X_2 = \text{NormaliseToUnitNorm}(X_2)$;
// Concatenate and apply PCA
$X_{\text{concat}} = \text{Concat}(X_1, X_2) \in \mathbb{R}^{n \times 2d}$;
$X = \text{PCA}(X_{\text{concat}}, d)$;

With this, we define a procedure (Algorithm 2) to combine four sets of embeddings by merging them in pairs first then merging the two results.

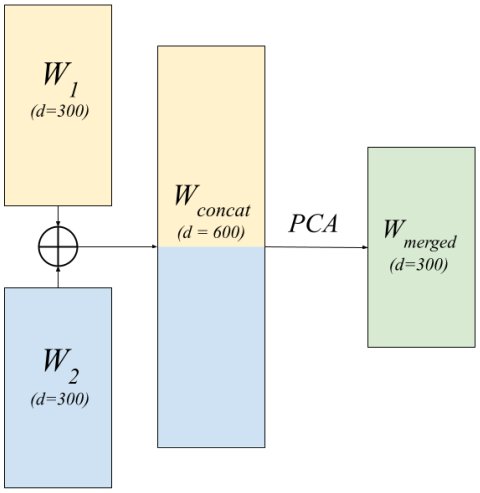

Figure 2: A visualization of the Merge procedure for obtaining a set of meta-embeddings from two separate sets of word embeddings. As detailed in algorithm 1, first the individual embeddings are concatenated and then dimension reduction via PCA is applied.

Algorithm 2: TripleMerge(X_i, X_o, Y_i, Y_o)

Data: Embedding matrices
$X_i, X_o, Y_i, Y_o \in \mathbb{R}^{n \times d}$
Result: A new meta-embedding $Z \in \mathbb{R}^{n \times d}$
$X_{\text{merged}} = \text{Merge}(X_i, X_o)$;
$Y_{\text{merged}} = \text{Merge}(Y_i, Y_o)$;
$Z = \text{Merge}(X_{\text{merged}}, Y_{\text{merged}})$;

Our final embeddings are defined as follows. We train one set of embeddings from the character 5-grams atomization. Hereby we refer to this model as "fastText" [3] and call its input and output embedding matrices FT_i, FT_o respectively. We then train another set of embeddings with the morphemes + stem (1-3)-grams atomization, which we refer to as "MorphoSeg". We label its input and output embedding matrices by MS_i, MS_o. Our final embeddings are the columns of the matrix $\text{TripleMerge}(\text{FT}_i, \text{FT}_o, \text{MS}_i, \text{MS}_o)$.

3.4 Analysis of rules-based segmenter

Here we discuss the strengths and weaknesses of the segmenter (introduced in section 3.1) as a core part of our methodology.

Strengths. We find that the segmenter performs well on and correctly identifies a wide range of

[3] This is similar but not identical to fastText, since we used another Wikipedia dump and used only character 5-grams and not (3-6)-grams due to computational constraints. However, we note that Grave et al. (2018) also changed the range from (3-6) to 5 and found it minimally impacted accuracy.

morphemes. In particular, it almost always correctly breaks down inflectional increments across morpheme boundaries, for instance with *marattai* 'tree(ACC)' $\longrightarrow$ *maram* 'tree' + *ai* (accusative suffix). Additionally, long agglutinative compounds are often broken up correctly, for instance *ezudappaṭukiṛadu* $\longrightarrow$ *ezuda* + *paṭu* + *kiṛa*+ *du*.

Weaknesses. However, we also find a number of distinct failure modes of the segmenter. We find undersegmentation of morphemes (e.g. inability to separate multiple stems in one word), oversegmentation of 'false' morphemes (in words that contain homophones of morphemes, e.g. *paccai* 'green' which happens to end in *ai*, the accusative suffix), ellision of certain morphemes, and difficulty with irregular forms. While it is beyond the scope of this paper to thoroughly analyze the segmenter, we anticipate that our gains on morphological tasks could be vastly improved with a better segmenter. More details are provided in our code.

4 Dataset

One of the primary contributions of this work is our novel Tamil analogy dataset, the first available for the language. The dataset is a hand-crafted set of 426 paired relations between words, and was produced by the authors. These word pairs are split into 10 semantic and 13 morphological relation types (see Appendix A). Analogy tetrads are then generated in a combinatorial fashion by combining pairs from the same class.

Given that Tamil is a low-resource language, automated construction of analogy dataset is relatively infeasible; lexicons rarely list fully inflected or complex morphological forms, and the use of a segmenter would subject the dataset to limitations similar to those discussed in section 3.4. As such, the decision to produce a human-curated analogy dataset was motivated by the desire to produce a gold-standard analogy task resource.

As a result of Tamil's morphological richness, even semantic relations often contain pairs with similar morphology. As such, we clustered word pairs within each relation type that shared identical morphology into labelled sub-classes. Analogies produced from morphologically identical pairs, along with analogies from morphological relations, were sorted into the **Subword** category of analogies, and semantic tetrads that were produced from morphologically non-identical pairs were placed in the **Non-subword** category.

Table 3 shows 4 examples of word tetrads for the semantic and morphological categories respectively, with two word pairs given per relation. The full dataset contains 4499 analogy tetrads, with 3487 analogy tetrads across 19 relation types in the **Subword** category and 1012 tetrads across 10 relation types in the **Non-subword** category. A complete list of relation types with examples (and numerical distributions across the full dataset, development, and test sets) can be found in Appendix A.

We note that our segmenter does not correctly identify all morphological relations. Therefore, MorphoSeg may not improve overall performance even in the subword category. However, we will see in Section 6 that it significantly improves performance on certain morphological relations, and that our final meta-embedding absorbs these strengths to substantially improve overall performance (across relations and categories).

5 Experimental setup

5.1 Training corpus and analogy task

We extracted [4] a corpus from the Tamil Wikipedia dump [5] (comprising 133732 articles) on April 20, 2020 and shuffled the sentences to obtain our final training corpus. A copy of the dump is available at our GitHub repository. We note that while Wikipedia may not capture the full range of Tamil inflectional morphology (having predominantly present/past tense and third-person conjugations), it captures rich derivational morphology that provides a good source of productive morphological diversity for our embeddings.

Our evaluation task measured performance on a word-analogy task performed by 'guessing' a missing word in our set of tetrads, which was computed by the gensim `most_similar` function (Řehůřek and Sojka, 2010). Correctness on each analogy tetrad was measured by top-k accuracy for each $k \in \{1, 5, 10\}$. From this, top-k accuracies were computed for each relation type. These accuracies were averaged within subword and non-subword categories, and overall model performance was measured by averaging the two figures. For brevity, we only present top-10 results here but we observe qualitatively similar behavior for top-1 and top-5 accuracy. Details are provided in Appendix C. We used a 75/25 dev/test split.

[4] https://github.com/attardi/wikiextractor
[5] https://dumps.wikimedia.org/

Type of relationship	Word Pair 1		Word Pair 2	
male- female	*āṇ* 'male'	*peṇ* 'female'	*rājā* 'king'	*rāṇi* 'queen'
profession- product	*kuyavar* 'potter'	*pānai* 'pot'	*necavāḷar* 'weaver'	*thuṇi* 'cloth'
animal- young	*nāi* 'dog'	*nāikkuṭṭi* 'dog-pup'	*pacu* 'cow'	*kanṛu* 'calf'
elder female kin- young male kin	*pāṭṭi* 'grandma'	*pēran* 'grandson'	*tāy* 'mother'	*makan* 'son'
nominative- accusative	*avan* 'he'	*avanai* 'him'	*kai* 'hand(nom)'	*kaiyai* 'hand(acc)'
adjective- adverb	*makizcciyāna* 'happy'	*makizcciyāka* 'happily'	*kopamāna* 'angry'	*kopamāka* 'angrily'
verb- for those who verb-ed	*vara* 'to come'	*vandavarkaḷukkāka* 'for those who came'	*pēca* 'to speak'	*pēciyavarkaḷukkāka* 'for those who spoke'
past- past completive COS	*keṭṭadu* 'it spoiled'	*keṭṭupōnadu* 'it spoiled'	*kuraindadu* 'it reduced'	*kuraindupōnadu* 'it reduced'

Figure 3: The table above shows four examples of tetrads from our **Non-Subword** analogy category, and four examples from our **Subword** category (in that vertical order). The table captures some of the variation inherent in Tamil verbal and noun forms even morphologically identical forms can vary in the way they append to a verbal/noun root (as in rows 7 and 8), and multiple morphemes often exist for a given meaning.

5.2 Implementation details

We used our training corpus to produce 300-dimensional embeddings. We based our training code on Tzu-Ray Su's PyTorch implementation of word2vec[6]. More hyperparameters are provided in Appendix B.

In our evaluation of our models, we were unable to utilize the full set of 4499 analogy tetrads, as many tetrads contained out-of-vocabulary (OOV) words due to the limitations of the Wikipedia training corpus. As our model was unable to did handle OOV tokens, we had to filter our dataset for applicable tetrads. After filtering for OOV tokens, there remained 1576 analogies (45.2%) across 13 relation types in the **Subword** category, and 794 analogies (78.5%) across 9 relation types in the **Non-subword** category.

We also trained a standard word2vec skip-gram model (Mikolov et al., 2013) as a baseline.

6 Results and Analysis

Results on the test set are shown in Figure 4. Our model was the strongest among those evaluated in both the subword and non-subword categories. First, we examine individual sets of word embeddings in section 6.1 and observe that they differ substantially in their success modes. In particular, we show that the incorporation of the morphological segmenter appears to significantly boost performance on certain morphological relations. Secondly, we turn to analyzing our meta-embeddings in section 6.2. Counterintuitively, we find that meta-embeddings in fact improve their performance when reduced to the same dimension as our original embeddings, seemingly combining the strengths of different representations.

6.1 Comparing individual models

'fastText, input' was the strongest individual model in both categories by a substantial margin. However, there are some relations in the dataset where it fell short and some of the other individual models performed better. As expected, 'MorphoSeg, input' was very effective on morphological relations that our segmenter correctly identified. More interestingly, both 'MorphoSeg, output' and 'fastText, output' performed better than 'fastText, input' across the kinship relations in the non-subword category. We hypothesize that kinship relations contain word pairs that rarely share subword information, so output vectors were more successful as they did not explicitly use subword-based atomization. Results on some such relations are shown in Figure 5.

[6] https://github.com/ray1007/pytorch-word2vec

Model	Subword	Non-subword	Average
Skip-gram, input	15.87	23.57	19.72
MS_i (MorphoSeg, input)	62.99	16.75	39.87
MS_o (MorphoSeg, output)	15.91	22.95	19.43
FT_i (fastText, input)	72.22	34.04	53.13
FT_o (fastText, output)	17.89	25.89	21.89
$\text{Concat}(\text{MS}_i, \text{MS}_o)^\dagger$	75.81	24.79	50.30
$\text{Merge}(\text{MS}_i, \text{MS}_o)$	83.07	29.99	56.53
$\text{Concat}(\text{FT}_i, \text{FT}_o)^\dagger$	72.69	42.68	57.68
$\text{Merge}(\text{FT}_i, \text{FT}_o)$	78.85	47.48	63.17
$\text{TripleMerge}(\text{FT}_i, \text{FT}_o, \text{MS}_i, \text{MS}_o)$	**90.24**	**48.52**	**69.38**

Figure 4: Top-10 accuracies of various models on our test set. The top row shows the standard skip-gram word2vec baseline. The next sub-table comprises the uncombined models arising from our atomization methods. The final sub-table consists of our final meta-embedding in the bottom row, as well as the intermediate meta-embeddings used to construct it (as described in algorithm 2). Models marked by † have 600 dimensions rather than 300 since they have only been concatenated.

Model	Nom-acc	Prof-prod	Kinship
MS_i	**80.15**	3.57	0.00
MS_o	20.59	17.86	17.86
FT_i	63.24	**35.71**	14.29
FT_o	21.32	25.00	**32.14**
TripleMerge	**88.97**	**64.29**	28.57

Figure 5: Top-10 accuracies of our four individual models and final meta-embeddings on a subset of relations. The final meta-embedding draws on complementary strengths of individual models to mitigate trade-offs between them.

This illustrates that the four individual embeddings had complementary success modes, suggesting the applicability of meta-embedding methods. Furthermore, the complementary strengths of models across relations appeared to occur along the lines of previously observed semantic-morphological trade-offs (Avraham and Goldberg, 2017), which warrants further investigation.

6.2 Improvements from meta-embeddings

The results highlight that both concatenation and PCA were highly effective in increasing performance. Each of $\text{Concat}(\text{MS}_i, \text{MS}_o)$ and $\text{Concat}(\text{FT}_i, \text{FT}_o)$ performed at least as well as each of their constituent individual models in both categories. Moreover, the PCA step (between Concat and Merge) consistently improved upon the Concat models by around 5% in both categories.

Examining the results of our final meta-embedding on each relation as in Figure 5 revealed that it drew on the complementary success modes of the individual models, thus mitigating the semantic-morphological trade-offs between them. In most relations, the meta-embedding performed at least as well as the best individual model, if not substantially better.

7 Conclusion and Future Work

In this paper, we investigated directly incorporating morphology into Tamil word embeddings using a morphological segmenter, following Bojanowski et al. (2017) in computing representations for subword units. We constructed a word analogy dataset for Tamil consisting of 13 types of morphological relations and 10 types of semantic relations to evaluate performance. We combine individual models to obtain more versatile meta-embeddings that seem to overcome previously observed trade-offs.

It remains for future work to investigate the performance of our techniques on OOV words, and the improvements better morphological segmentation might bring. Evaluating our embeddings on other tasks for Indian languages, such as Akhtar et al. (2017)'s Tamil word similarity dataset, remains an important direction, as does studying the importance of incorporating morphology for downstream tasks such as POS tagging and NMT (Kumar et al., 2020). Exploring the applicability of our pipeline of morphological segmentation and meta-embeddings to other morphology-rich languages is another avenue for future work.

References

Mostafa Abdou, Artur Kulmizev, and Vinit Ravishankar. 2018. Mgad: Multilingual generation of analogy datasets. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Syed Sarfaraz Akhtar, Arihant Gupta, Avijit Vajpayee, Arjit Srivastava, and Manish Shrivastava. 2017. Word similarity datasets for Indian languages: Annotation and baseline systems. In *Proceedings of the 11th Linguistic Annotation Workshop*, pages 91–94, Valencia, Spain. Association for Computational Linguistics.

Oded Avraham and Yoav Goldberg. 2017. The interplay of semantics and morphology in word embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 422–426, Valencia, Spain. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Cristian Cardellino. 2016. Spanish billion words corpus and embeddings (march 2016). *URL http://crscardellino. me/SBWCE*.

Joshua Coates and Danushka Bollegala. 2018. Frustratingly easy meta-embedding – computing meta-embeddings by averaging source word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 194–198, New Orleans, Louisiana. Association for Computational Linguistics.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1287–1292, Denver, Colorado. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1).

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA).

Peng Jin and Yunfang Wu. 2012. Semeval-2012 task 4: evaluating chinese word similarity. In ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 374–377.

Arun Kumar, Ryan Cotterell, Lluís Padró, and Antoni Oliver. 2017. Morphological analysis of the Dravidian language family. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 217–222, Valencia, Spain. Association for Computational Linguistics.

Saurav Kumar, Saunack Kumar, Diptesh Kanojia, and Pushpak Bhattacharyya. 2020. "a passage to India": Pre-trained word embeddings for Indian languages. In *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pages 352–357, Marseille, France. European Language Resources association.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, Sofia, Bulgaria. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Jiaqi Mu and Pramod Viswanath. 2018. All-but-the-top: Simple and effective postprocessing for word representations. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Yirong Pan, Xiao Li, Yating Yang, and Rui Dong. 2020. Morphological word segmentation on agglutinative languages for neural machine translation. *CoRR*, abs/2001.01589.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. pages 141–150.

Vikas Raunak, Vivek Gupta, and Florian Metze. 2019. Effective dimensionality reduction for word embeddings. In *Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019*, pages 235–243. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Wikipedia. 2020. Tamil language — Wikipedia, the free encyclopedia. [Online; accessed 12-May-2020].

Peter Wittek, Bevan Koopman, Guido Zuccon, and Sándor Darányi. 2013. Combining word semantics within complex hilbert space for information retrieval. In *Quantum Interaction - 7th International Conference, QI 2013, Leicester, UK, July 25-27, 2013. Selected Papers*, volume 8369 of *Lecture Notes in Computer Science*, pages 160–171. Springer.

Wenpeng Yin and Hinrich Schütze. 2016. Learning word meta-embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1351–1360, Berlin, Germany. Association for Computational Linguistics.

A Dataset Details

In this section of the Appendix, we provide details of our word analogy dataset and its construction. Essentially, as mentioned in the main work, we present a set of 4499 word tetrads split across 10 semantic and 13 morphological categories.

A.1 Relation types

In our first section here, we provide examples of each relation type we generated words for, with illustrative examples provided in Tamil text, Roman transliteration, and translations. Tables 1 and 2 show semantic and morphological word pair categories respectively.

A.2 Distribution of pairs across relation types

In Tables 3 and 4, we show the numerical distribution of pairs across categories in the full dataset and the dataset used in the paper for evaluation after filtering of OOV tokens. We attempt to attain as even a spread as possible over analogy categories (and furnish a range of morphology over the language).

A.3 Distribution of tetrads across relation types

In Tables 5 and 6, we show the numerical distributions of analogy tetrads over our distinct relation types. We also provide here the final train/test split of our data to show that the relative distributions over relation types were largely maintained, and also seek to show the breakdown of analogies across relation types in the original, unfiltered dataset. We review here briefly the process by which analogy tetrads are constructed. Within our 10 semantic and 13 morphological relation types, we assign pairs with similar morphology to a class. Following this, word pairs are combinatorially paired to produce tetrads; pairs of the same class and the same relation type, that is, pairs that share identical/highly similar morphology get assigned to the **Subword** class of analogies, and tetrads consisting of two divergent pairs get assigned to the **Non-subword** class. The idea here is that we want to differentiate analogies that can be solved with use of subword information from those that cannot as such, we notate this in our results, and capture this distinction across our different relation types in Tables 5 and 6.

B Implementation Details

B.1 Atomization

There is a subtle difference between the n-grams we take of the entire word in the character 5-grams atomization and the n-grams we take of the stem in the morphemes + stem (1-3)-grams atomization. Specifically, Tamil is an abugida script, which means vowel values attached to consonants are expressed as a series of diacritics.

The original fastText paper and the character 5-grams method we implemented separate diacritics from their stem consonants since they are given distinct Unicode characters. However, for the morphemes + stem (1-3)-grams, we tried taking n-grams with and without separate diacritics and stems, and ultimately chose not to separate them. We used a smaller n-gram window of 1-3 to account for this.

B.2 Training Hyperparameters

We tabulate all hyperparameters in Table 7. These were mostly unchanged from the defaults used in Tzu-Ray Su's original GitHub repository. The only change was that we used 5 negative samples, following the original fastText setup (Bojanowski et al., 2017).

B.3 Alternative Dimension Reduction Methods

We briefly note that we tried incorporating the dimension reduction method proposed by Raunak et al. (2019), which combines removing the top few principal components of the embeddings with PCA. We found that this was less effective than simple PCA for our embeddings. We hypothesize that this was because our embeddings did not have disproportionately high top singular values, contrasting the observations made by Raunak et al. (2019) for the embeddings that they considered.

C Detailed Results

This section expands on the results presented in the body of this paper in two ways: we show top-k results for each $k \in \{1, 5, 10\}$, and we show these results for each individual relation in our dataset.

We note that we attempted to compare our models against other existing baselines trained on slightly different corpora such as those released by Kumar et al. (2020). However, due to the different corpora, these models had additional OOV tokens in our analogy dataset that we would have had to

remove to evaluate them together with our models. Running methods such as ours alongside many standard baselines on a fixed corpus and comparing the resulting models is an important area for future work.

C.1 Results in subword categories

Results for top-1, top-5, and top-10 accuracies are shown in Tables 8, 9, and 10 respectively. Categories are numbered according to the numbering convention established in Tables 1 and 2.

The TripleMerged model (our final set of meta-embeddings) generally outperforms all other models in these categories by margins of $\approx 10\%$, although this is not uniformly true across all categories. This is to be expected since this is the only meta-embedding incorporating both the FT_i and MS_i embeddings, which are the two individual embeddings that incorporate subword information. This explanation is also supported by the strong performances of these two individual embeddings in the Subword categories.

C.2 Results in non-subword categories

Results for top-1, top-5, and top-10 accuracies are shown in Tables 11, 12, and 13 respectively. The strongest performing models here are TripleMerged, $\text{FT}_{\text{concat}}$, and $\text{FT}_{\text{merged}}$. While TripleMerged generally outperforms $\text{FT}_{\text{concat}}$, $\text{FT}_{\text{merged}}$ is in general slightly better than TripleMerged in these categories. This is once again to be expected since the FT_i and FT_o models are the best-performing individual embeddings in the Non-subword categories. Still, it is remarkable that TripleMerged is only slightly worse in general than $\text{FT}_{\text{merged}}$, given that it was the result of merging $\text{FT}_{\text{merged}}$ with $\text{MS}_{\text{merged}}$, which was significantly weaker in the Non-subword categories.

C.3 Overall results

Overall results averaged across categories are shown in Table 14. TripleMerged exhibits the strongest performance overall, compensating for its slight weakness in the Non-subword category with significant improvements in the Subword category on all other models.

No.	Category	Example	Transliteration	Meaning
0	male-female	ராஜா ராணி	*rājā rāṇi*	'king/queen'
1	me-my	அவன் அவனுடைய	*avan avanuḍaiya*	'he/his'
2	profession-product	நெசவாளர் துணி	*necavāḷar thuṇi*	'weaver/cloth'
3	fruit_A-tree_A	கொய்யா கொய்யாமரம்	*koyyā koyyāmaram*	'guava/guava-tree'
4	verb_form-noun_form	யோசிக்க யோசனை	*yōcikka yōcanai*	'to think/thought'
5	animal-young	மாடு கன்று	*pacu kanṛu*	'cow/calf'
6	kin_elder-kin_young[1]	மாமா மருமகன்	*māmā marumakan*	'maternal uncle/nephew'
7	kin_elder-kin_young[2]	தந்தை மகள்	*thanthai makaḷ*	'father/daughter'
8	kin_elder-kin_young[3]	பாட்டி பேரன்	*pāṭṭi pēran*	'grandmother/grandson'
9	positive-negative	வெற்றி தோல்வி	*verṛi thōlvi*	'victory/loss'

Table 1: This table shows our numbered relation types for semantic word pairs, consisting of 10 types. We provide Tamil text, transliteration, and translations.

No.	Category	Example	Transliteration	Meaning
10	nom-acc	அவன் அவனை	*avan avanai*	'he/him'
11	nom-dat	அவள் அவளுக்கு	*avaḷ avaḷukku*	'she/to her'
12	adjective-adverb	மகிழ்ச்சியான மகிழ்ச்சியாக	*makizcciyāna makizcciyāka*	'happy/happily'
13	verb-past_respect	பறக்க பறந்தார்	*paṛakka paṛaṅthār*	'fly/they flew'
14	past-past_cmp1_inan	மாறியது மாறிபோனது	*māṛiyathu māṛipōnathu*	'changed/changed(completive)'
15	verb-verb_doer_dat	பார்க்க பார்த்தவர்களுக்காக	*pṛkka pṛtthavarkaḷukkāka*	'to see/for the sake of those who saw'
16	past-past_cmp2_male	பண்ணான் பண்ணிவிட்டான்	*paṇṇān paṇṇivittān*	'he did/he did(completive)'
17	past-past_cmp2_female	பண்ணாள் பண்ணிவிட்டாள்	*paṇṇāḷ paṇṇivittāḷ*	'she did/he did(completive)'
18	male_past-female_past	பண்ணான் பண்ணாள்	*paṇṇān paṇṇāḷ*	'he did/she did'
19	verb-doer_female	பாட பாடியவள்		'to sing/the female who sang'
20	verb-doer_male	பாட பாடியவன்	*pāṭa pāṭiyavan*	'to sing/the male who sang'
21	verb-passive_pl_inan	காண காணப்பட்டன	*kāṇa kāṇappaṭṭana*	'to see/they were seen'
22	verb-past_inan	சொல்ல சொன்னது	*colla connadu*	'to say/that which was said'

Table 2: This table shows our numbered relation types for morphological word pairs, consisting of 13 types. We provide Tamil text, transliteration, and translations.

Dataset	Semantic relation type										Total
	0	1	2	3	4	5	6	7	8	9	
Full dataset	29	30	9	9	23	5	8	4	4	24	145
OOV Removed	22	30	8	6	22	5	8	4	4	22	131

Table 3: Numerical distribution of analogy pairs across the different semantic relation types (see Figure for individual relation type meanings/examples

Dataset	Morphological relation type													Total
	10	11	12	13	14	15	16	17	18	19	20	21	22	
Full dataset	19	19	22	30	22	15	20	20	30	19	19	23	23	281
OOV Removed	17	19	20	27	0	0	0	1	10	0	1	16	17	128

Table 4: Numerical distribution of analogy pairs across the different morphological relation types (see Figure for individual relation type meanings/examples

Dataset		Semantic relation type										Total
		0	1	2	3	4	5	6	7	8	9	
Full dataset	Subword	43	354	1	12	62	3	0	0	0	5	480
	Non-subword	363	81	35	24	191	7	28	6	6	271	1012
Development set	Subword	5	265	0	4	39	2	0	0	0	3	318
	Non-subword	168	60	21	6	134	5	21	4	4	169	592
Test set	Subword	2	89	0	2	13	1	0	0	0	2	109
	Non-subword	56	21	7	3	45	2	7	2	2	57	202

Table 5: Numerical distribution of analogy tetrads across the different semantic relation types (see Figure for individual relation type meanings/examples

Dataset		Morphological relation type													Total
		10	11	12	13	14	15	16	17	18	19	20	21	22	
Full dataset	Subword	171	171	231	435	231	105	190	190	435	171	171	253	253	3007
Development set	Subword	102	128	142	263	0	0	0	0	33	0	0	90	102	860
Test set	Subword	34	43	48	88	0	0	0	0	12	0	0	30	34	289

Table 6: Numerical distribution of analogy tetrads across the different morphological relation types (see Figure for individual relation type meanings/examples

Epochs	Initial LR	LR schedule	Momentum	Batch size	Context window	Negative samples
5	0.025	Linear annealing	0.0	100	5	5

Table 7: Training hyperparameters that we used.

Model	sub_0	sub_1	sub_3	sub_4	sub_5	sub_9	sub_10	sub_11	sub_12	sub_13	sub_18	sub_21	sub_22	sub_mean
skipgram_input	12.50	12.36	0.00	1.92	0.00	0.00	13.97	12.79	14.06	6.25	2.08	1.67	3.68	6.25
MS_input	0.00	**71.07**	**50.00**	19.23	25.00	12.50	32.35	37.79	58.33	21.31	22.92	30.00	20.59	30.85
MS_output	37.50	11.52	0.00	0.00	0.00	0.00	8.82	12.79	9.38	5.97	0.00	2.50	0.74	6.86
FT_input	25.00	6.18	25.00	30.77	50.00	25.00	11.76	17.44	48.44	22.44	85.42	**74.17**	8.09	33.05
FT_output	25.00	14.04	0.00	0.00	25.00	0.00	10.29	11.63	6.77	6.82	2.08	4.17	4.41	8.48
MS_concat	12.50	53.09	12.50	7.69	0.00	25.00	36.76	38.37	66.15	32.67	25.00	30.83	25.74	28.18
MS_merged	25.00	64.61	0.00	9.62	50.00	37.50	**54.41**	49.42	76.56	41.19	31.25	47.50	38.24	40.41
FT_concat	**50.00**	28.37	12.50	21.15	0.00	37.50	19.85	23.84	47.40	25.57	**89.58**	60.00	8.09	32.60
FT_merged	**50.00**	28.09	12.50	30.77	25.00	**87.50**	22.79	34.30	60.42	32.95	**89.58**	69.17	17.65	43.13
TripleMerged	**50.00**	62.36	0.00	**32.69**	**75.00**	62.50	44.85	**52.91**	**80.73**	**57.95**	66.67	72.50	**58.09**	**55.10**

Table 8: Detailed per-category top-1 accuracies in the Subword category for all models we considered.

Model	sub_0	sub_1	sub_3	sub_4	sub_5	sub_9	sub_10	sub_11	sub_12	sub_13	sub_18	sub_21	sub_22	sub_mean
skipgram_input	12.50	28.37	0.00	5.77	0.00	0.00	27.21	24.42	22.92	16.19	6.25	5.00	11.03	12.28
MS_input	25.00	80.90	**62.50**	53.85	25.00	37.50	64.71	58.72	96.88	44.60	54.17	55.83	58.82	55.27
MS_output	37.50	21.63	0.00	0.00	0.00	0.00	18.38	25.58	15.62	14.77	2.08	3.33	5.15	11.08
FT_input	**75.00**	37.08	**62.50**	46.15	75.00	**87.50**	35.29	44.19	84.38	43.47	**97.92**	83.33	22.79	61.12
FT_output	37.50	31.18	0.00	1.92	25.00	12.50	19.12	25.00	12.50	12.22	10.42	5.83	5.88	15.31
MS_concat	62.50	80.06	50.00	44.23	75.00	25.00	69.85	62.79	92.71	61.36	50.00	65.00	60.29	61.45
MS_merged	62.50	83.43	50.00	63.46	75.00	75.00	77.21	68.02	96.88	70.17	50.00	83.33	70.59	71.20
FT_concat	62.50	58.99	25.00	38.46	**100.00**	**87.50**	49.26	54.65	80.73	53.98	**97.92**	80.83	26.47	62.79
FT_merged	**75.00**	62.64	37.50	50.00	**100.00**	**87.50**	54.41	61.05	92.71	64.77	**97.92**	90.00	52.94	71.26
TripleMerged	**75.00**	**87.08**	50.00	**75.00**	**100.00**	62.50	**81.62**	**74.42**	**98.44**	**87.22**	85.42	**95.83**	**80.88**	**81.03**

Table 9: Detailed per-category top-5 accuracies in the Subword category for all models we considered.

Model	sub_0	sub_1	sub_3	sub_4	sub_5	sub_9	sub_10	sub_11	sub_12	sub_13	sub_18	sub_21	sub_22	sub_mean
skipgram_input	12.50	35.39	0.00	7.69	0.00	12.50	32.35	30.23	25.52	21.31	8.33	5.00	15.44	15.87
MS_input	37.50	85.67	62.50	61.54	25.00	50.00	80.15	70.35	97.92	51.70	56.25	74.17	66.18	62.99
MS_output	37.50	27.25	0.00	1.92	25.00	12.50	20.59	30.23	19.79	18.47	2.08	4.17	7.35	15.91
FT_input	75.00	61.80	**75.00**	51.92	75.00	**100.00**	63.24	58.14	95.31	58.52	**97.92**	85.83	41.18	72.22
FT_output	37.50	37.36	0.00	3.85	25.00	12.50	21.32	28.49	17.71	17.61	12.50	9.17	9.56	17.89
MS_concat	75.00	85.39	**75.00**	67.31	75.00	75.00	82.35	75.58	98.44	75.85	56.25	76.67	67.65	75.81
MS_merged	**87.50**	87.92	62.50	76.92	**100.00**	75.00	**88.97**	**81.98**	99.48	85.23	56.25	95.83	82.35	83.07
FT_concat	62.50	69.38	50.00	46.15	**100.00**	87.50	62.50	66.86	95.31	66.48	**97.92**	86.67	53.68	72.69
FT_merged	**87.50**	75.56	37.50	59.62	**100.00**	87.50	72.79	69.19	98.44	74.43	**97.92**	93.33	71.32	78.85
TripleMerged	**87.50**	**91.29**	**75.00**	**88.46**	**100.00**	87.50	**88.97**	79.65	**100.00**	**92.05**	91.67	**99.17**	**91.91**	**90.24**

Table 10: Detailed per-category top-10 accuracies in the Subword category for all models we considered.

Model	nonsub_0	nonsub_1	nonsub_2	nonsub_3	nonsub_4	nonsub_5	nonsub_6	nonsub_7	nonsub_8	nonsub_9	nonsub_mean
skipgram_input	25.00	7.14	14.29	0.00	1.67	0.00	**14.29**	0.00	0.00	10.09	7.25
MS_input	0.00	25.00	3.57	**8.33**	2.78	0.00	0.00	0.00	0.00	0.00	3.97
MS_output	26.34	10.71	7.14	0.00	1.11	0.00	7.14	0.00	25.00	10.53	8.80
FT_input	12.95	2.38	0.00	**8.33**	8.33	0.00	0.00	**12.50**	12.50	9.65	6.66
FT_output	24.55	11.90	3.57	0.00	2.22	0.00	7.14	0.00	25.00	10.09	8.45
MS_concat	12.50	30.95	7.14	0.00	6.67	0.00	0.00	0.00	0.00	7.89	6.52
MS_merged	13.84	34.52	10.71	0.00	11.11	12.50	0.00	0.00	0.00	8.33	9.10
FT_concat	30.80	9.52	14.29	0.00	7.22	12.50	7.14	0.00	**37.50**	14.47	13.35
FT_merged	**33.93**	9.52	**28.57**	0.00	11.11	12.50	7.14	**12.50**	**37.50**	**20.61**	**17.34**
TripleMerged	21.88	**36.90**	25.00	0.00	**19.44**	**25.00**	3.57	**12.50**	0.00	17.11	16.14

Table 11: Detailed per-category top-1 accuracies in the Non-subword category for all models we considered.

Model	nonsub_0	nonsub_1	nonsub_2	nonsub_3	nonsub_4	nonsub_5	nonsub_6	nonsub_7	nonsub_8	nonsub_9	nonsub_mean
skipgram_input	35.71	27.38	17.86	0.00	3.89	0.00	21.43	25.00	37.50	16.23	18.50
MS_input	4.02	64.29	3.57	16.67	16.67	12.50	0.00	0.00	0.00	2.63	12.03
MS_output	34.82	20.24	14.29	0.00	3.33	0.00	7.14	25.00	**50.00**	12.72	16.75
FT_input	28.12	13.10	28.57	**33.33**	23.33	**25.00**	0.00	12.50	37.50	18.86	22.03
FT_output	35.71	33.33	14.29	0.00	8.33	12.50	**28.57**	25.00	37.50	13.60	20.88
MS_concat	22.77	72.62	14.29	8.33	20.56	12.50	0.00	0.00	12.50	14.91	17.85
MS_merged	23.66	**77.38**	17.86	8.33	30.00	**25.00**	3.57	12.50	12.50	18.86	22.97
FT_concat	45.09	35.71	35.71	25.00	21.11	**25.00**	25.00	37.50	**50.00**	28.07	32.82
FT_merged	**46.43**	38.10	**53.57**	25.00	27.22	**25.00**	21.43	**50.00**	**50.00**	**30.26**	**36.70**
TripleMerged	41.52	75.00	**53.57**	0.00	**39.44**	**25.00**	21.43	37.50	25.00	26.75	34.52

Table 12: Detailed per-category top-5 accuracies in the Non-subword category for all models we considered.

Model	nonsub_0	nonsub_1	nonsub_2	nonsub_3	nonsub_4	nonsub_5	nonsub_6	nonsub_7	nonsub_8	nonsub_9	nonsub_mean
skipgram_input	43.30	40.48	25.00	0.00	4.44	0.00	28.57	37.50	37.50	18.86	23.57
MS_input	7.59	82.14	3.57	25.00	30.56	12.50	0.00	0.00	0.00	6.14	16.75
MS_output	40.62	34.52	17.86	0.00	5.00	12.50	17.86	25.00	**62.50**	13.60	22.95
FT_input	40.62	39.29	35.71	**41.67**	27.78	**50.00**	14.29	25.00	37.50	28.51	34.04
FT_output	37.95	38.10	25.00	0.00	11.67	12.50	32.14	37.50	50.00	14.04	25.89
MS_concat	30.36	86.90	17.86	16.67	36.67	12.50	3.57	12.50	12.50	18.42	24.79
MS_merged	32.59	90.48	39.29	8.33	42.22	25.00	10.71	12.50	12.50	26.32	29.99
FT_concat	50.00	46.43	46.43	**41.67**	29.44	25.00	32.14	**75.00**	50.00	30.70	42.68
FT_merged	**56.25**	50.00	57.14	33.33	41.67	25.00	**39.29**	**75.00**	**62.50**	**34.65**	47.48
TripleMerged	51.34	**91.67**	**64.29**	25.00	**52.22**	**50.00**	28.57	62.50	25.00	**34.65**	**48.52**

Table 13: Detailed per-category top-10 accuracies in the Non-subword category for all models we considered.

Model	Top-1			Top-5			Top-10		
	Subword	Non-subword	Overall	Subword	Non-subword	Overall	Subword	Non-subword	Overall
skipgram_input	6.25	7.25	6.75	12.28	18.50	15.39	15.87	23.57	19.72
MS_input	30.85	3.97	17.41	55.27	12.03	33.65	62.99	16.75	39.87
MS_output	6.86	8.80	7.83	11.08	16.75	13.92	15.91	22.95	19.43
FT_input	33.05	6.66	19.86	61.12	22.03	41.58	72.22	34.04	53.13
FT_output	8.48	8.45	8.46	15.31	20.88	18.10	17.89	25.89	21.89
MS_concat	28.18	6.52	17.35	61.45	17.85	39.65	75.81	24.79	50.30
MS_merged	40.41	9.10	24.75	71.20	22.97	47.08	83.07	29.99	56.53
FT_concat	32.60	13.35	22.97	62.79	32.82	47.81	72.69	42.68	57.68
FT_merged	43.13	**17.34**	30.24	71.26	**36.70**	53.98	78.85	47.48	63.17
TripleMerged	**55.10**	16.14	**35.62**	**81.03**	34.52	**57.78**	**90.24**	**48.52**	**69.38**

Table 14: Average top-1, top-5, and top-10 accuracies for all models in the Sub-word and Non-subword categories, as well as overall accuracies (taken to be the average of Sub-word and Non-subword scores).