Scaling Neural ITN for Numbers and Temporal Expressions in Tamil: Findings for an Agglutinative Low-resource Language

Bhavuk Singhal
Uniphore Inc.
bhavuk.singhal@uniphore.com

Sindhuja Gopalan
Uniphore Inc.
sindhuja@uniphore.com

Amrith Krishna
Uniphore Inc.
krishnamrith12@gmail.com

Malolan Chetlur
Uniphore Inc.
malolan.chetlur@uniphore.com

Abstract

Inverse Text Normalization (ITN) involves rewriting the verbalised form of text from spoken transcripts to its corresponding written form. The task inherently expects challenges in identifying ITN entries due to spelling variations in words arising out of dialects, transcription errors etc. Additionally, in Tamil, word boundaries between adjacent words in a sentence often get obscured due to *Punarchi*, i.e. phonetic transformation of these boundaries. Being morphologically rich, the words in Tamil show a high degree of agglutination due to inflection and clitics. The combination of such factors leads to a high degree of surface-form variations, making scalability with pure rule-based approaches difficult. Instead, we experiment with fine-tuning three pre-trained neural LMs, consisting of a seq2seq model (s2s), a non-autoregressive text editor (NAR) and a sequence tagger + rules combination (tagger). While the tagger approach works best in a fully-supervised setting, s2s performs the best (98.05 F-Score) when augmented with additional data, via bootstrapping and data augmentation (DA&B). S2S reports a cumulative percentage improvement of 20.1 %, and statistically significant gains for all our models with DA&B. Compared to a fully supervised setup, bootstrapping alone reports a percentage improvement as high as 14.12 %, even with a small seed set of 324 ITN entries.

1 Introduction

Inverse Text Normalisation (ITN) is a text-rewriting approach that converts the verbalized form of text in spoken conversational systems to its written form.[1] The written and verbalised forms often diverge in their surface-forms (van Esch and Sproat, 2017; Sproat et al., 2001).Such words, henceforth to be referred to as ITN entities, typically include numbers, dates, money, etc. At large, such categories are referred to as semiotic classes (Taylor, 2009). ITN is generally perceived as a task for improving text-readability for any language (Sunkara et al., 2021). However, recent research suggests that identifying ITN entities may additionally improve the performance of systems designed for downstream NLU tasks (Thawani et al., 2021; Pouran Ben Veyseh et al., 2020; Sundararaman et al., 2022). In this work, we identify and address the challenges in developing ITN systems for a low-resource and morphologically rich agglutinative language, Tamil.

We primarily consider four different categories of ITN entities, in our task. Three of them are numerals belonging to semiotic classes (Sproat et al., 2001) and the fourth one is linguistic phrases denoting temporal expressions. The three numerical categories are MONEY, DATE AND TIME, and OTHER numerical values. TEMPORAL expressions, though typically do not require a rewrite, are also considered as an additional category in our task. For instance, consider the statement 'I will pay by the end of this month'. While the temporal expression 'end of this month' may not require a rewrite for readability, the information it conveys is similar to that one would expect from ITN entities, such as '30th May'. Hence, such expressions are also identified, for further downstream processing.

Tamil is a morphologically-rich agglutinative classical Dravidian language, widely spoken in India, Sri Lanka, etc. (Eberhard et al., 2022; Kolipakam et al., 2018). Table 1 demonstrates various such affixation for 'muppattañcu', the written form of the numeral 35. Here, rows 7-8 and 9-10 in Table 1 portray affix synonymy (Deo, 2007). Further, the boundaries between the adjacent words in a Tamil sentence may be obscured due to *Punarchi*, owing to phonetic transformations at these boundaries (Sastri, 1934). The obscured word boundaries may

[1] In speech systems, text normalisation (TN) is a step performed prior to text synthesis, where written form of the text is converted to its verbalised form. ITN does the inverse of TN, typically during speech recognition and hence named accordingly.

Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 440–450

Sl. No	**Inflected form**	**English Translation**	**Morphological Segmentation**
1	muppattañcuṅkaḷā	is it thirty-five	muppattañcu + ṅkaḷā
2	muppattañcukku	for thirty-five	muppattañcu + kku
3	muppattañcil	in thirty-five	muppattañcu + il
4	muppattañcukkuḷḷa	within thirty-five	muppattañcu + kkuḷḷa
5	muppattañcuvāṭṭi	thirty-five times	muppattañcu + vāṭṭi
6	muppattañcukkumkūṭa	even at thirty-five	muppattañcu + kkumkūṭa
7	muppattañcume	all of thirty-five	muppattañcu + me
8	muppatañcaiyum		muppatañcu + aiyum
9	muppattañcām	thirty-fifth	muppattañcu + ām
10	muppattañcāvatu		muppattañcu + āvatu

Table 1: Surface-forms due to Inflection and Clitic for 'muppattañcu', the written form for 35,

lead to ambiguity in identifying individual words from a joint form. An ITN entity may undergo *Punarchi* with other unrelated non-ITN words. Consider a Tamil sentence, "Inta mātattavaṇai muppattorāyirattiyĕṭnūtti muppattañciruvā".[2] Here, the phrase 'inta māta' (this month) is an ITN entity. However, the word 'māta' (month) undergoes *Punarchi* with an unrelated word 'tavaṇai' (installment) to form 'mātattavaṇai'[3]. Hence, the word inta and the substring 'māta' from 'mātat**t**avaṇai' needs to be identified as the ITN entity.

Identifying temporal expressions poses several challenges. One, temporal expressions may affect other related words in a sentence (Vashishtha et al., 2020), such as the tense of the verb. Further, a temporal expression may be represented as multiple words, with unrelated words appearing in between the expression. Hence, a single entity may be formed by multiple non-contiguous spans. Consider the sentence 'nāl̇aikku kālaila nīṅka añcu maṇikkul̇̇la iṅka vantuṭuṅka'.[4] Here, the word 'nīṅka' (you) appears between four words that collectively represent a single temporal expression 'nāl̇aikku' (tomorrow) 'kālaila' (morning) 'añcu maṇikku' (5 'O clock). The combination of Punarchi, inflection, clitics, dialects, and potential transcription errors make identifying ITN entities a challenging task in Tamil.

ITN systems typically are developed using rule-based systems (Neubig et al., 2012), neural text rewriting methods (Zhang et al., 2019), or a combination of neural taggers followed by rule-based methods (Tan et al., 2023). A purely rule-based approach may be challenging for Tamil, due to the aforementioned characteristics of the language. Hence, we explore three different neural approaches, a sequence-to-sequence model (Xue et al., 2022), a non-auoregressive text-editor model (Mallinson et al., 2020) and a combination of neural tagger (Conneau et al., 2020) with rules.

We leverage pretrained large language models for fine-tuning these models for our task. However, Tamil is a low-resource language. Hence, we additionally explore data augmentation and bootstrapping to obtain additional data to train our models. Specifically, we perform data augmentation by obtaining a substitution matrix of common spelling variations, generating verbalised forms of numerals, and identifying temporal expressions from publicly available corpora. For bootstrapping, our default setting involves a human-in-the-loop (HitL) approach for candidate verification at each iteration. We compare the default setting with two other experimental setups, a) a fully automated setup replacing HitL with a number classifier, and b) a warm start scenario with a seed set many times larger than the default setup.

Our major observations from this work are:

1. The seq2seq model reports the best overall score with an F-Score of 98.05, a 2.06 % increase from that of the tagger+rules system. However, in the fully supervised setting, and also with bootstrapping, the tagger+rules system outperformed others.

2. Bootstrapping, irrespective of the three variants, reports significant performance improvements for all three models, compared to a fully supervised setup. The percentage improvements range from 9.13 to 14.12 %.

[2]translation: 'This month's instalment is ₹31,835.'

[3]māta + tavaṇai → mātat**t**avaṇai

[4]translation: 'You may come here at 5 AM tomorrow morning'.

Input	inta mātattavaṇai muppattorāyiratti ĕṭṇūtti muppattañciruvā
Final output from system	{Inta māta}# [this month] ttavaṇai {muppattorāyiratti ĕṭṇūtti muppattañciruvā} [₹31,835]#
Final sentence with improved readability	Inta mātattavaṇai ₹31,835

Table 2: Sample input along with the corresponding system output and the sentence with improved readability for all our proposed models.

3. Data-augmentation alone contributes to more than half of our training data. It leads to statistically significant improvements. Seq2seq reports the highest gain and the tagger reports the lowest, with percentage improvements of 5.24 % and 0.74 % respectively on top of the gains made on bootstrapping.

2 ITN Models

ITN is a monotone sequence transduction task where the input and output sequences typically have considerable lexical overlap and generally follow monotonicity in their alignments (Schnober et al., 2016; Krishna et al., 2018). Here, we formulate the task in three different setups. a) A sequence tagger (Conneau et al., 2020) coupled with a rule-based system; b) A seq2seq model (Xue et al., 2022; Raffel et al., 2020); c) A non-autoregressive text-editor (Mallinson et al., 2020)

Table 2 shows a sample input sequence, previously discussed in Section 1. Irrespective of the setup we use, the input and outputs do not change, though there may be intermediary outputs depending on the systems involved in each setup. We focus not only on improving text readability but also to identify ITN entities for downstream processing. Hence, the 'final output from the system' contains both the verbalised forms as well as the corresponding rewrites generated. Moreover, the verbalised form of the ITN entities is enclosed within the '{' and '}' markup. Similarly, its corresponding rewrite is generated and enclosed within the '[' and ']' markup. Non-ITN words are devoid of any markups. Finally, those markup blocks suffixed with a '#', along with non-ITN words, remain in the 'final sentence with improved readability'.

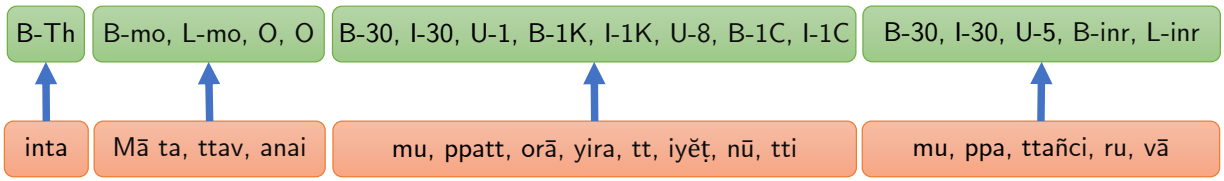
2.1 Sequence-Tagger with Rules

We first identify text spans that form ITN entities and then perform deterministic rule-based transformations based on the label set of the tagger. We follow a tagging scheme inspired by the IOBES and BILOU scheme (Ratinov and Roth, 2009; Lester, 2020) for our tags. We altogether have a label set of 94 labels, 47 of them are used for representing temporal expressions and the rest are used for representing the other three numeral categories.

Figure 1 illustrates the tagging sequence for a given input sentence. Here, non-entity tokens (subwords) are tagged with the O tag. Since we need the entity tags for rewrite, we need to identify the exact values of the numerals involved and that can potentially lead to an infinite set of possible values. Hence, the numeral entities are decomposed into sub-units. We consider each whole number from 0 to ten as a separate label. Further, place values from units to a trillion, and additionally place values adopted in the Indian numbering system such as 'lakh' (hundred thousand) and 'crore' (ten million) are also considered.

Consider the verbalised form of ₹31,835, which is 'muppattorāyirattiyĕṭnūtti muppattañciruvā'. 31,835 is decomposed[5] into multiple units and its verbalised form is tagged with the labels 30, 1, 1K, 8, 1C, 30, and 5. Here, 1K and 1C are place values denoting a thousand and a hundred respectively. Inspired by the BILOU scheme, The first token of each unit is prefixed with 'B-', any interior token of a unit is prefixed with 'I-', and a token that fully covers a sub-unit is prefixed with 'U-'. The last token of a whole entity is prefixed with 'L-', though the final token of each sub-unit is not separately marked. Finally, there may be subwords that overlap the text portion of two separate units due to Punarchi. For instance, 'iyĕṭ' in Figure 1 represents one such case where the 'i' is part of 'āyiratti' (thousand place value, 1K) and 'y' is the common string created due to Punarchi and the remaining is part of the string representing 8. For the token, we assign it the label 'U-8', as otherwise, there would be representation for the number 8 in the sequence.

[5] $(30 + 1) \times 1000 + 8 \times 100 + 30 + 5$



Input: inta mātattavaṇai muppattorāyiratti ĕṭṇūtti muppattañciruvā

Figure 1: Tagger output for the sub-word tokens of the input sequence.

2.2 Non-autoregressive Text Editor

We follow FELIX (Mallinson et al., 2020; Rothe et al., 2021), a non-autoregressive text editor model. It consists of a tagging model and an insertion model both of which can be trained independently. Given an input sequence **s**, the corresponding final output sequence from the system **y** is generated based on the conditional probability: $p(\mathbf{y}|\mathbf{s}) = p_{ins}(\mathbf{y}|\mathbf{y}^m)p_{tag}(\mathbf{y}^t|\mathbf{s})$

Here, $\mathbf{y}^t$ is the output of the tagging model. **y** is the final output from the system based on a masked sequence $\mathbf{y}^m$ as input to the insertion model. The masked sequence is determined using the predictions from the tagging model. The tagger predicts the labels to either retain (R) or delete (D) the tokens. Further, a source token is tagged either with an R-Ins^K or D-Ins^K, where the 'R'/'D' in it is the decision for the current token and K is the number of tokens to insert after it.

Figure 2 shows the predictions from the tagging model, i.e. $\mathbf{y}^t$. Based on the tags in $\mathbf{y}^t$, a sequence $\mathbf{y}^m$ is obtained. Here, those tokens tagged with R and R-Ins^K are retained. The tokens tagged with D and D-Ins^K are also made part of $\mathbf{y}^m$ but are enclosed within a special marker to indicate that those tokens need to be deleted. Finally, depending on the value of Ks predicted, the corresponding number of $MASK$ tokens are also inserted into $\mathbf{y}^m$. The sequence corresponding to the 'Final output from system' row in Table 2 is then generated by the insertion model based on $\mathbf{y}^m$ as input.

2.3 Seq2Seq model

We use a standard auto-regressive formulation, maximising the output sequence likelihood with teacher forcing (Sutskever et al., 2014; Cao et al., 2021). Here, similar to FELIX, we directly predict the desired written form, the final output from the system as shown in Table 2.

3 Dataset Generation

Tamil being a low resource language, we employ both bootstrapping (Yarowsky, 1995) and data augmentation (Feng et al., 2021) for obtaining the training data for the task.

3.1 Bootstrapping

Expanding from a small seed set of ITN entities we iteratively create labeled instances from a large set of unlabelled ASR transcriptions. The seed set is ensured to contain at least one verbalised for each of the 102 labels (§2.1), such that these can be combined to form complex ITN entities.

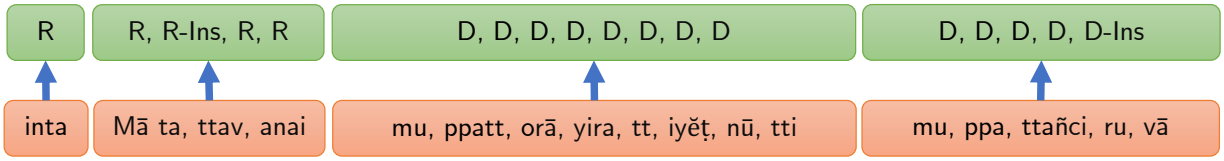
Approximate string matching approaches such as Jaro (Jaro, 1989) and Jaro-Winkler (Winkler, 1990; Cohen et al., 2003) are used to expand our seed set. Matching words are then validated either using a human-in-the-loop (HitL) approach or with a fully automated approach using a classifier. For the latter, a numeral classifier is built that learns to identify verbalised forms of text belonging to valid numerals Johnson et al. (2020).

3.2 Data-Augmentation[6]

To enrich our training dataset, we utilize data-augmentation techniques in three key areas. To handle **spelling variations** in transcripts caused by transcription errors, agglutination, and Punarchi, we create a substitution matrix of character n-grams (up to 3-grams) based on matched entity pairs from bootstrapping. **Numerals** are augmented using Tamildict[7], with suffixes added based on the substitution matrix for proper date/time formats. We introduce sentences with **temporal expressions** by aligning corresponding Tamil phrases using a multilingual word aligner (Jalili Sabet et al., 2020).

[6]We elaborate on each of the following augmentation strategies in the appendix (§A.2)

[7]https://www.tamildict.com/



Input: inta mātattavaṇai muppattorāyiratti ĕṭṇūtti muppattañciruvā

Figure 2: Tag sequence $\mathbf{y}^t$ from the tagging component of the text-editor for the input sequence.

4 Experiments

4.1 Data collection[8]

For bootstrapping, we collect 203,187 raw-ASR transcriptions from code-mixed Tamil-English telephonic conversations, with 22.7 % token share in English. Here, we utilize publicly available resources for data generation (§3;), such as Tamildict, Shabdkosh[9], Encyclopedia[10], Tyagi et al. (2021), Kakwani et al. (2020), Ramesh et al. (2022) .

Seed Set: By default, we assume a cold-start HitL bootstrap setting, where the seed set contains 324 commonly used ITN entries in Tamil and English. These entries are obtained based on the labels based on inputs from native speakers. Additionally, we experiment with a warm start scenario with 10,000 ITN words, including entries from the cold-start setting, Tyagi et al. (2021), and the rest from Tamildict.

Train splits: The training dataset consists of 30,417 sentences exactly matching the (cold-start) seed ITN phrases, 24,632 additional sentences from HitL bootstrapping, and 66,713 sentences from data augmentation, including temporal expressions and numerals.

Validation and Test Splits: We use 2,000 sentences for validation and 5,000 sentences for the test split, which are verified and corrected by Tamil speakers.

Number Classifier: Using warm-start seed set and equivalent negatives, we train the classifier. Cold start: 3,623 ITN-entries, 28,520 additional sentences. Warm start: 8,129 ITN-entries, 60,948 additional sentences.

Metrics: We evaluate using micro-averaged Precision, Recall, and F1-Score. Edit-distance based word-error rates are assessed for ITN entities and non-ITN words (Sunkara et al., 2022) using the 'Final output from system' (Table 2).

System	P	R	F	IW	NW
Tagger	97.46	94.72	96.07	5.69	0.42
NAR	93.29	91.62	92.45	8.28	0.59
S2S	**98.62**	**97.48**	**98.05**	**2.46**	**0.18**

Table 3: Overall results for ITN

Systems: XLM-Roberta is fine-tuned for the tagger (**Tagger**), Mallinson et al. (2020) for the text editor (**NAR**), and byT5 (Xue et al., 2022) for seq2seq setup (**S2S**). We use XLM-Roberta as the base model for the NAR (Mallinson et al., 2020)

4.2 Results[11]

Table 3 shows the performance of all the three systems we consider. S2S currently outperforms others in all the metrics. We find that NAR performs worse than both the other systems. It reports an entity F-Score of 92.45 as against that of 98.05 and 96.07 from the S2S and the tagger respectively. When predicting entities, all the systems report higher precision than recall scores. Given the diverse decoding strategies adopted in these systems, we also compare the error rates between the final predicted sequence and the ground truth sequence. S2S remains closest to the ground truth, both in the prediction of ITN entities and the non-ITN words with an I-WER (IW) of 2.46 and N-WER (NW) of 0.18 respectively. Table 3 also shows the IW and NW for all three systems.

Category Level Predictions: Table 4 reports the category-level performance (F-Score) of all the systems we consider in this work. S2S and Tagger report the best and second-best scores respectively in all four categories. Here, all the systems report their lowest scores in the Money category. We observe that among the mispredictions in the Money

[8] Additional details for our data collection process is elaborated in the appendix (§B)

[9] https://www.shabdkosh.com/

[10] https://omniglot.com/writing/tamil.htm

[11] Following Dror et al. (2018), we perform pairwise t-tests and observe that all the scores reported are statistically significant ($p < 0.05$) unless otherwise stated

Category	Tagger	NAR	S2S
MONEY	92.72	89.38	**97.13**
DATE AND TIME	96.65	92.79	**98.81**
OTHER Numerals	96.13	93.67	**98.63**
TEMPORAL	95.59	93.39	**97.82**

Table 4: Category level results (F-Score)

Size	Tagger	NAR	S2S
Exact Matching 30K	**87.39**	82.58	81.64
Bootstrapping +24K = 54K	**95.37**	90.28	93.17
Data Augmentation +66K = 120K	96.07	92.45	**98.05**

Table 5: Results (F-Score) by incrementally adding data via bootstrapping and data-augmentation.

category, 56.79 % of those get mispredicted to the other Numerals category. NAR performs the best in predicting 'Other Numerals', in relative comparisons to its performance on other categories. Similarly, both S2S and Tagger tend to perform the best in predicting the 'Date and Time' category compared to other classes.

Impact of Bootstrapping and Augmentation: While Tagger reports the best scores in the fully supervised setup and in bootstrapping, S2S reports the overall best score (98.05) after data augmentation. As observed in several other tasks that use encoder-decoder models (Gu et al., 2018), we hypothesise that the increased data due to augmentation leads to improvements for the S2S model. As shown in Table 5, all the systems improve their performance after both bootstrapping and data augmentation. Here, S2S reports the highest percentage improvement after both steps. It has a percentage improvement of 14.12 after incorporating Bootstrapping and a further percentage improvement of 5.24 after data augmentation.

Bootstrapping Setups: Table 6 reports the performance of all the systems in 3 bootstrapping setups.[12] Exact matching even with the large warm-start seed set of 10,000 entries reports an F-Score of only 88.42 for the tagger, highest of the three systems. However, even a cold start fully-automated 'classifier' setup in bootstrapping, reports significant improvements to all the models with 89.36

[12]None of the setups include data from data-augmentation

System	Cold Start		Warm Start
	Human-in-the-Loop	Classifier	Classifier
Tagger	95.37	94.49	95.63
NAR	90.28	89.36	91.92
S2S	93.17	92.54	95.09

Table 6: Results (F-Score) for the three different bootstrapping setups

being the lowest F-Score reported (for NAR). HitL setup, our default configuration, reports statistically significant gains compared to the 'classifier' setup in the cold-start setting. Here, tagger's performance in the 'classifier' setup, decreased to an F-Score of 94.49, from 95.37, and that of the S2S decreased to 92.54, from 93.17. However, in the warm-start setting, the 'classifier' setup surpasses the HitL cold-start setup. Here, both NAR and S2S leads to statistically significant improvements whereas tagger reports a higher score, though not statistically significant.

5 Conclusion

Our work focuses on developing a neural ITN system for a morphologically-rich agglutinative language, Tamil. Tamil is a morphologically productive language with rich agglutination, which along with Punarchi leads to high degree of surface-form variation in the utterances generated. We observe that both bootstrapping and data-augmentation for data generation help improve the performance of all the three systems we experimented with. S2S reports the highest gain. It surpasses tagger and reports the best score, when using data from data generation. Without data augmentation, Tagger reports the best scores in all the other settings. Even in a cold start setting, we observe that a fully automated candidate verification can lead to performance improvements in these models. However, our HitL cold start setting or alternatively the fully automated solution in the warm start setting has shown to further improve the performance of these models. Overall, we find that both seq2seq and tagger models perform satisfactorily for our use cases and helps in downstream applications.

Limitations

Our work's scope is currently focused on a limited set of semiotic classes, three of those focusing specifically on numerals. In future, we would like

to expand to other semiotic classes such as abbreviations and acronyms. Similarly, we currently focus only on text-rewriting and identification of ITN entries. However, we believe joint modelling of other related tasks such as grammatical and spelling error correction, punctuation restoration etc. may benefit the performance of all the tasks. We leave this for future work.

References

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *International Conference on Learning Representations*.

William W Cohen, Pradeep Ravikumar, and Stephen E Fienberg. 2003. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the 2003 International Conference on Information Integration on the Web*, pages 73–78.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Ashwini Deo. 2007. Derivational morphology in inheritance-based lexica: Insights from pāṇini. *Lingua*, 117(1):175–201.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

David M. Eberhard, Gary F. Simons, and Charles D. Fennig. 2022. *Ethnologue: Languages of the World*, 25 edition. SIL International, Dallas.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988, Online. Association for Computational Linguistics.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. 2018. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 344–354, New Orleans, Louisiana. Association for Computational Linguistics.

Masoud Jalili Sabet, Philipp Dufter, François Yvon, and Hinrich Schütze. 2020. SimAlign: High quality word alignments without parallel training data using static and contextualized embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1627–1643, Online. Association for Computational Linguistics.

Matthew A Jaro. 1989. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420.

Devin Johnson, Denise Mak, Andrew Barker, and Lexi Loessberg-Zahl. 2020. Probing for multilingual numerical understanding in transformer-based language models. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 184–192, Online. Association for Computational Linguistics.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, Online. Association for Computational Linguistics.

Vishnupriya Kolipakam, Fiona M Jordan, Michael Dunn, Simon J Greenhill, Remco Bouckaert, Russell D Gray, and Annemarie Verkerk. 2018. A bayesian phylogenetic study of the dravidian language family. *Royal Society open science*, 5(3):171504.

Amrith Krishna, Bodhisattwa P. Majumder, Rajesh Bhat, and Pawan Goyal. 2018. Upcycle your OCR: Reusing OCRs for post-OCR text correction in Romanised Sanskrit. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 345–355, Brussels, Belgium. Association for Computational Linguistics.

Brian Lester. 2020. iobes: Library for span level processing. In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, pages 115–119, Online. Association for Computational Linguistics.

Jonathan Mallinson, Aliaksei Severyn, Eric Malmi, and Guillermo Garrido. 2020. FELIX: Flexible text editing through tagging and insertion. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1244–1255, Online. Association for Computational Linguistics.

Graham Neubig, Yuya Akita, Shinsuke Mori, and Tatsuya Kawahara. 2012. A monotonic statistical machine translation approach to speaking style transformation. *Computer Speech Language*, 26(5):349–370.

Amir Pouran Ben Veyseh, Franck Dernoncourt, Quan Hung Tran, and Thien Huu Nguyen. 2020. What does this acronym mean? introducing a new dataset for acronym identification and disambiguation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3285–3301, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Gowtham Ramesh, Sumanth Doddapaneni, Aravinth Bheemaraj, Mayank Jobanputra, Raghavan AK, Ajitesh Sharma, Sujit Sahoo, Harshita Diddee, Mahalakshmi J, Divyanshu Kakwani, Navneet Kumar, Aswin Pradeep, Srihari Nagaraj, Kumar Deepak, Vivek Raghavan, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh Shantadevi Khapra. 2022. Samanantar: The largest publicly available parallel corpora collection for 11 indic languages. *Transactions of the Association for Computational Linguistics*, 10:145–162.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado. Association for Computational Linguistics.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. A simple recipe for multilingual grammatical error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 702–707, Online. Association for Computational Linguistics.

PS Subrahmanya Sastri. 1934. *History of Grammatical Theories in Tamil and their relation to the Grammatical Literature in Sanskrit*. Journal of Oriental Research.

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. 2016. Still not there? comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1703–1714, Osaka, Japan. The COLING 2016 Organizing Committee.

Richard Sproat, Alan W Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. 2001. Normalization of non-standard words. *Computer Speech and Language*, 15(3):287–333.

Dhanasekar Sundararaman, Vivek Subramanian, Guoyin Wang, Liyan Xu, and Lawrence Carin. 2022. Improving downstream task performance by treating numbers as entities. In *Proceedings of the 31st ACM International Conference on Information amp; Knowledge Management*, CIKM '22, page 4535–4539, New York, NY, USA. Association for Computing Machinery.

Monica Sunkara, Chaitanya Shivade, Sravan Bodapati, and Katrin Kirchhoff. 2021. Neural inverse text normalization. In *ICASSP 2021*.

Srinivas Sunkara, Maria Wang, Lijuan Liu, Gilles Baechler, Yu-Chung Hsiao, Jindong Chen, Abhanshu Sharma, and James W. W. Stout. 2022. Towards better semantic understanding of mobile interfaces. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5636–5650, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Sharman Tan, Piyush Behre, Nick Kibre, Issac Alphonso, and Shuangyu Chang. 2023. Four-in-one: a joint approach to inverse text normalization, punctuation, capitalization, and disfluency for automatic speech recognition. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 677–684. IEEE.

Paul Taylor. 2009. *Text-to-speech synthesis*. Cambridge university press.

Avijit Thawani, Jay Pujara, Filip Ilievski, and Pedro Szekely. 2021. Representing numbers in NLP: a survey and a vision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–656, Online. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Shubhi Tyagi, Antonio Bonafonte, Jaime Lorenzo-Trueba, and Javier Latorre. 2021. Proteno: Text normalization with limited data for fast deployment in text to speech systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers*, pages 72–79, Online. Association for Computational Linguistics.

Daan van Esch and Richard Sproat. 2017. An expanded taxonomy of semiotic classes for text normalization. *Proc. Interspeech 2017*, pages 4016–4020.

Siddharth Vashishtha, Adam Poliak, Yash Kumar Lal, Benjamin Van Durme, and Aaron Steven White. 2020. Temporal reasoning in natural language inference. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4070–4078, Online. Association for Computational Linguistics.

William E Winkler. 1990. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Hao Zhang, Richard Sproat, Axel H. Ng, Felix Stahlberg, Xiaochang Peng, Kyle Gorman, and Brian Roark. 2019. Neural models of text normalization for speech applications. *Computational Linguistics*, 45(2):293–337.

A Dataset Generation

In this appendix, we provide additional details regarding the generation of the data discussed in Section 3.

A.1 Detailed Bootstrapping Process

To obtain the training data, we employ bootstrapping, combining it with data augmentation for better coverage. Initially, we curate a seed set of ITN entities that contains all the basic forms of entities required for the task. We also collect synonyms and paraphrases for non-numeral entries in the seed set. To identify spelling mistakes, inflectional variants, and Punarchi-related variations, we use approximate string-matching, including Jaro and Jaro-Winkler similarities.

In the bootstrapping process, we match seed entries with text spans in the transcripts using Jaro and Jaro-Winkler similarities, generating two sorted lists of top-matching entries. The candidates from these lists need to be filtered based on their validity before adding them to the seed set for the next iteration. We have two filtering options: the human-in-the-loop (HitL) approach and the classifier-based automated step.

In the HitL approach, candidates are verified manually by a Tamil speaker and then added to the seed set. For the automated step, we build a numeral classifier that identifies verbalized forms of valid numerals. We use a feed-forward classifier with pretrained embeddings to encode the input. Training data for the classifier consists of valid numeral sequences as positive examples and other verbalized text forms as negative examples. Additionally, we generate invalid numeral sequences as further negative examples.

By employing bootstrapping and data augmentation, we iteratively expand the seed set and obtain a large labeled dataset for training our sequence tagger.

A.2 Detailed Data-Augmentation

In this appendix, we provide a comprehensive explanation of the methodologies and implementation details for each data-augmentation technique used in our research.

Spelling Variations: Spelling variations in transcripts, encompassing transcription errors, agglutination variations, and Punarchi effects, can significantly influence the performance of language models. For addressing these variations, we employ a substitution matrix approach. We delve into the creation of the character n-gram substitution matrix, explaining how it is derived from entity pairs matched during the bootstrapping process. Furthermore, we describe the alignment of character n-grams and the aggregation process to identify the most likely substitutes.

Generating Numerals: Numerals are an essential component of many linguistic tasks. We present the process of generating numerals using the Tamildict[13] resource and demonstrate how we incorporate them into transcript sentences containing other numerals. The addition of appropriate suffixes based on the substitution matrix is explained in detail, as well as the constraints we implement to ensure proper date and time formats.

Temporal Expressions: To augment our dataset with sentences containing temporal expressions, we elaborate on our approach using publicly available corpora. We discuss the collection of common temporal expressions in Tamil and English and provide insights into extracting relevant sentences from the corpora. Additionally, we delve into the alignment of English-Tamil sentence pairs using a multilingual word aligner (Jalili Sabet et al., 2020), en-

[13] https://www.tamildict.com/

suring the extraction of aligned and contextually relevant temporal expressions in Tamil.

By providing detailed methodologies in the appendix, readers can gain deeper insights into our data-augmentation techniques and understand their impact on improving the quality and effectiveness of our trained language model.

B Data Collection for Bootstrapping

In this appendix, we provide additional details regarding data collection discussed in Section 4.1.

For bootstrapping our ITN extraction training data, we collected a total of 203,187 raw-ASR transcriptions from an in-house speech collection. These transcriptions are derived from code-mixed Tamil-English telephonic conversations, with a token share of 22.7% in English. To further enhance the dataset, we utilized various publicly available resources, including:

- Tamil Text-Normalization Corpus: We leveraged a Tamil text-normalization corpus (Tyagi et al., 2021) to obtain additional data for our task.
- Unlabelled Tamil Corpus: Another publicly available unlabelled Tamil corpus (Kakwani et al., 2020) was utilized for data generation.
- Parallel Tamil-English Corpus: We incorporated data from a parallel Tamil-English corpus (Ramesh et al., 2022) to augment our dataset.
- Tamil and English Dictionaries: We utilized resources such as Tamildict[14], Shabdkosh[15], and Encyclopedia[16] to enrich the data.

Seed Set Curation: For our cold-start scenario, we curated a seed set containing 324 commonly used ITN entries in both Tamil and English. This seed set was carefully verified using the aforementioned dictionaries and encyclopedias. In the warm-start scenario, we expanded the seed set to include 10,000 ITN words. This larger set included the initial 324 entries from the cold-start setting, 6,163 entries from Tyagi et al. (2021), and the rest from Tamildict.

[14] https://www.tamildict.com/
[15] https://www.shabdkosh.com/
[16] https://omniglot.com/writing/tamil.htm

Training Data Generation: The training dataset for our ITN extraction task was constructed in multiple steps:

- We obtained 30,417 sentences (30K) that exactly matched the seed ITN phrases.
- The HitL bootstrapping approach resulted in an additional 24,632 sentences (24K) extracted from the raw-ASR transcriptions.
- Through bootstrapping, we identified 27.58% additional entities in the existing 30K sentences.
- Data augmentation further contributed 66,713 sentences, with 19,709 of them containing temporal expressions, and 9,608 sentences containing both temporal expressions and numerals. Sentences with temporal expressions were sourced from Kakwani et al. (2020) and Ramesh et al. (2022), while sentences with numerals were obtained from Tyagi et al. (2021). Additionally, numerals were generated using Tamildict and incorporated into existing sentences.

Number Classifier: To enhance our ITN extraction training dataset, we utilize a number classifier. The classifier is trained using the warm-start seed set along with an equivalent number of negative examples. In the cold start setting, it identifies 3,623 ITN entries, while in the warm start setting, it identifies 8,129 additional ITN entries.

Evaluation Metrics: For evaluating our ITN extraction models, we use micro-averaged entity-level Precision (P), Recall (R), and F1-Score (F), which are commonly used metrics in Named Entity Recognition (NER) setups (Tjong Kim Sang and De Meulder, 2003). Additionally, we calculate edit-distance based word-error rates separately for ITN entities (IW) and non-ITN words (NW) (Sunkara et al., 2022). These metrics provide a comprehensive assessment of the model's performance.

Experimental Systems: In our experiments, we employ three different systems for ITN extraction. First, we fine-tune XLM-Roberta (Conneau et al., 2020) using a rules setup for the tagger (**Tagger**). Second, we use the approach proposed by Mallinson et al. (2020) for the text editor (**NAR**), where XLM-Roberta serves as the tagger. Finally, we utilize byT5 (Xue et al., 2022) for the seq2seq

setup (**S2S**). For the text-editor, we utilize XLM-Roberta as the tagger to facilitate ITN extraction. The training hyperparameters for both models can be found in Table 7. Default values were utilized for all other hyperparameters.

Hyperparameters	**XLM-Roberta**	**ByT5**
Maximum Sequence Length	150	450
Batch Size	100	24
Learning Rate	1e-4	5e-4
Epochs	80	4

Table 7: Hyperparameters