A Computational Analysis of Vagueness in Revisions of Instructional Texts

Alok Debnath
Kohli Center for Intelligent Systems
International Institute of Information Technology, Hyderabad
alok.debnath@research.iiit.ac.in

Michael Roth
Institute for Natural Language Processing
University of Stuttgart
michael.roth@ims.uni-stuttgart.de

Abstract

WikiHow is an open-domain repository of instructional articles for a variety of tasks, which can be revised by users. In this paper, we extract pairwise versions of an instruction before and after a revision was made. Starting from a noisy dataset of revision histories, we specifically extract and analyze edits that involve cases of vagueness in instructions. We further investigate the ability of a neural model to distinguish between two versions of an instruction in our data by adopting a pairwise ranking task from previous work and showing improvements over existing baselines.

1 Introduction

Instructional texts aim to describe the actions necessary to accomplish a task or goal, in as clear and concise a manner as possible. *WikiHow*[1] is an extensive compendium of instructional guides for various topics and domains. Any user may edit the articles, and *WikiHow* collates these revision histories. The edit history of such informal instructional articles is a source of user-generated data that can help identify possible reasons and necessities for editing. *wikiHowToImprove* (Anthonio et al., 2020) is a dataset that compiles revision histories for the analysis of linguistic phenomena that occur in edits of instructional texts, ranging from the correction of typos and grammatical errors to the clarification of ambiguity and vagueness.

In this paper, we focus on cases of lexical vagueness, defined as "lexeme[s] with a single but non-specific meaning" (Tuggy, 1993), which can potentially cause misunderstandings in instructional texts. Specifically, we study vagueness based on the change in the main verb in the original and revised version of an instruction. We say that an instruction was vague if, upon revision, the revised main verb is contextually more specific than the original version. Some examples of vague and clarified instructions are provided in Table 1. As indicated by the examples, the revised verb is usually more specific in that it provides additional information on how or why an action needs to be taken.

Original Sentence	Revised Sentence
Then, **make** the floor and walls of your house.	Then, **design** the floor and walls of your house.
When you **go** to the Hogwarts park...	When you **visit** the Hogwarts park...
Get a flexible single cord.	**Purchase** a flexible single cord.

Table 1: Examples of vague instructions and their more clarified versions from the *wikiHowToImprove* Dataset

The classification of vague and clarified instructions is a first step towards automatic text editing for clarification based on linguistic criteria such as ambiguity and vagueness at a sentence level. Existing tools for text editing focus on text simplification and fact editing (Malmi et al., 2019), while others are designed for grammatical error correction (Xie et al., 2018). Our work acts as the first step towards automated editing based on linguistic criteria by identifying vague instructions and differentiating them from "clarified" ones. Our use of the *wikiHowToImprove* corpus also utilizes a resource of edit pairs, therefore introducing a new dataset for the linguistic study of vagueness as well as exploring the general versatility of such corpora.

Our contributions are to create a dataset of vague and clarified instructions, provide an analysis based on semantic frames, and demonstrate the first results of a neural model's ability to distinguish the two versions. We create and analyze the dataset by extracting relevant instances from *wikiHowToIm-*

[1] https://www.wikihow.com/

Proceedings of the 16th Conference of the European Chapter of the Associationfor Computational Linguistics: Student Research Workshop, pages 30–35
April 19 - 23, 2021.

prove, using POS tags, dependency features, and edit distance as constraints, as well as FrameNet frames as features (Section 3). We then devise a pairwise ranking task, where we train and evaluate different neural models and analyze their performance based on frame relations and differences in distributional word representations (Section 4).

2 Related Work

Our paper focuses on revisions in wikiHow for a specific linguistic phenomenon, namely vagueness. The motivation to use revision histories as corpora for NLP tasks was introduced by Ferschke et al. (2013). The task of defining and categorizing edit intentions has been explored well for the Wikipedia edits corpus (Yang et al., 2016, 2017). More recently, Anthonio et al. (2020) performed a similar categorization on the revisions in *WikiHow*.

Traditional computational analyses of vague statements have been based on logical representations (DeVault and Stone, 2004; Tang, 2008). In contrast, our focus is on vagueness in terms of lexical changes in revisions, which is more similar to previous analyses that considered the context-dependent resolution of vague expressions such as colour references (Meo et al., 2014). Other computational approaches to vagueness include, the detection of vague sense definitions in ontological resources (Alexopoulos and Pavlopoulos, 2014) and website privacy policies (Lebanoff and Liu, 2018) as well as the verification of historical documents (Vertan, 2019).

Our approach to identifying and classifying vagueness is analyzed using FrameNet frames which provide specialized relations among conceptual categories, in a manner similar to recent advances in neural models that use sentence-level information to perform hyponymy–hypernymy classification. Roller et al. (2018) analyzes lexico-syntactic pattern-based instances of word-specific hypernymy-hyponymy constructions. Snow et al. (2004) explores the extraction of predefined patterns for hypernyms and hyponyms in the same sentence, while Shwartz et al. (2016) incorporates distributional methods for their classification using sentence-level features.

3 Data Creation, Preprocessing, and Analysis

WikiHow articles mostly contain instructions, but also include descriptions, explanations, and other non-instructional sentences that provide additional context. The *wikiHowToImprove* corpus (Anthonio et al., 2020) is an unfiltered corpus of revision histories. Therefore, we first need to extract those revisions where the original and revised versions are both instructional sentences, which can be done based on syntactic properties (§3.1). We then use a FrameNet parser to determine the frames (and their relationships) evoked by the root verb in the original and revised version of an instruction (§3.2).

The final extracted data consists of only those revisions where the root verb has been modified to be more specific to the sentence. This extracted corpus consists of 41,615 sentences.

3.1 Data Extraction and Cleaning

wikiHowToImprove is a noisy source of data with misspellings, non-standard abbreviations, grammatical errors, emoticons, etc. In order to use the data for our task, we first perform some cleaning and preprocessing.

We filter the typos and misspellings in the dataset by comparing all the vocabulary words to words in the English dictionary using the *Enchant* python API[2]. After filtering the typos, we POS tag and dependency parse the data using the *Stanza* library[3] (Qi et al., 2020). We discard all sentence pairs where the sentences are shorter than four or longer than 50 words.

We then create a sub-corpus of instructional sentneces by extracting those edit pairs in which both the original and revised version of a sentence fulfill at least one of the following criteria:

- imperative form—the root verb has no nominal subject (e.g. “Please finish the task”);
- instructional indicative form—the nominal subject of the root verb is ‘you,’ ‘it’ or ‘one’ (e.g. “You should finish the task”);
- passive form with ‘let’—the sentence is in passive voice, and the root verb is ‘let’ (e.g. “Let the paper be put on the table.”).

Finally, we retain only those sentence pairs whose character edit distance is smaller than 10. This filter was added after empirical tests to accommodate changes in the verb form and syntactic frame while ensuring that there are little to no additional edits (often just vandalism or spam).

[2] https://pyenchant.github.io/
[3] https://stanfordnlp.github.io/stanza/

3.2 Verb Frame Analysis

We perform an analysis of verb frame relations from this extracted corpus using the FrameNet hierarchy (Baker et al., 1998). In order to identify evoked frames from the data, we use the INCEpTION Project's neural *FrameNet Tools* parser[4] (Klie et al., 2018; Markard and Klie, 2020). FrameNet Tools identifies the frame-evoking elements, the evoked frames, and the context elements' roles in these frame for a given sentence. In this work, we ignore role assignments and only consider predictions of evoked frames, which we found to be generally reliable in our data.[5]

We extract the frame of the root verb in the original and revised sentences. For each pair, we identify the frame relation, if any, using the NLTK FrameNet API[6](Schneider and Wooters, 2017). We found that most edits could be categorized into one of the following frame relations between the frames evoked by the original and revised verb frames:

1. **Subframe-of**: The original frame refers to a complex scenario that consists of several individual states, one of which is the revised frame. (e.g. TRAVERSING→ARRIVING: "*Go* to the thumbs up log." is revised to "*Visit* the thumbs up log.")

2. **Inherits-from**: The frame of the revised verb elaborates on the frame evoked by the original verb (e.g., DECIDING→CHOOSING: "*Determine* the card you want to buy" is revised to "*Choose* which card you want to buy.")

3. **Uses**: The frame of the revised verb uses or weakly inherits properties of the original verb frame (e.g., PERCEPTION_ACTIVE→SCRUTINY: "*Look* for the best fit for your taste" is revised to "*Search* for the best fit for your taste.").

We also find cases of contextually relevant clarifications for phrasal verbs, such as "*Make* your bed" vs. "*Fix* your bed..." which are not covered in FrameNet. Further, there are cases in which the FrameNet Tools parser did not identify the main verb or could not assign a frame. For instance, the verb *compel* as in "you may feel *compelled* ..." is not in FrameNet. We categorize these instances, which are fewer in number than the other categories, under a single **Other** category and leave further inspection to future work. A distribution of instances over categories is shown in Table 2. Apart from instances from the 'Other' category, we indeed found the main verbs in the revised versions of a sentence to be more specific than in the original versions.

Relation	**Total**	**Train**	**Test**	**Val**
Usage	15,243	11,084	2,194	1,965
Inheritance	13,166	9,179	2,008	1,793
Subframe	9,481	6,835	1,720	926
Other	3,925	2,833	649	443
Total	41,615	30,044	6,237	5,334

Table 2: Number of sentences in the extracted dataset and distribution of FrameNet relations between original and revised verbs. We also show the distribution of train, test and validation for each frame relation.

4 Pairwise Ranking Experiments

In this section, we investigate if a neural model can distinguish between the original and revised version of the same instruction. We describe a neural architecture that uses a joint representation designed for comparing two versions of a sentence before predicting an output. We compare our results to a standard BiLSTM-Attention model used in previous work (Anthonio et al., 2020).

4.1 System and Training Details

The initial components of our system are two BiLSTM modules, LSTM_{1A} and LSTM_{1B}, that each takes one version of a sentence as input. The individual BiLSTMs are followed by a joint layer LSTM_{AB} and an additional layer of BiLSTM modules, LSTM_{2A} and LSTM_{2B}, that re-encode the sentence based on the joint representations. The final layer is trained to predict for each sentence, whether it is the original or revised version, labeling them 0 or 1, respectively.

In practice, we first encode versions A and B of an instruction using FastText embeddings or BERT. The embedded sentences S_A and S_B are then passed through LSTM_{1A} and LSTM_{1B} one (sub-word) token at a time. The hidden layers $\mathbf{h}_{1A}$

[4] https://github.com/inception-project/framenet-tools

[5] Although automatic frame identification is noisy, the tools used here are implementations of the unimodal model presented in Botschen et al. (2018), which achieves a high accuracy of over 88%.

[6] http://www.nltk.org/howto/framenet.html

and $\mathbf{h}_{1B}$ are then concatenated and passed through LSTM_{AB}, whose output $\mathbf{h}_{AB}$ is then concatenated again with the original hidden states to re-encode each sentence version in LSTM_{2A} and LSTM_{2B}. Lastly a classification layer, trained using a cross-entropy objective, transforms the final representations $\mathbf{h}_{2A}$ and $\mathbf{h}_{2B}$ into a real-valued output score using self-attention, which is normalized by softmax and rounded to $\{0, 1\}$. The equations below give a simplified summary of our implementation.[7]

$$\mathbf{h}_{1A} = \text{LSTM}_{1A}(S_A) \quad (1)$$

$$\mathbf{h}_{1B} = \text{LSTM}_{1B}(S_B) \quad (2)$$

$$\mathbf{h}_{AB} = \text{LSTM}_{AB}(\mathbf{h}_{1A} \cdot \mathbf{h}_{1B}) \quad (3)$$

$$\mathbf{h}_{2A} = \text{LSTM}_{2A}(\mathbf{h}_{AB} \cdot \mathbf{h}_{1A}) \quad (4)$$

$$\mathbf{h}_{2B} = \text{LSTM}_{2B}(\mathbf{h}_{AB} \cdot \mathbf{h}_{1B}) \quad (5)$$

$$l_A = \left[\frac{\exp(\mathbf{w}^\top \mathbf{h}_{2A})}{\exp(\mathbf{w}^\top \mathbf{h}_{2A}) + \exp(\mathbf{w}^\top \mathbf{h}_{2B})} \right] \quad (6)$$

$$l_B = \left[\frac{\exp(\mathbf{w}^\top \mathbf{h}_{2B})}{\exp(\mathbf{w}^\top \mathbf{h}_{2A}) + \exp(\mathbf{w}^\top \mathbf{h}_{2B})} \right] \quad (7)$$

Training Details We experiment with both FastText (Grave et al., 2018) and BERT (Devlin et al., 2019), using representations with a dimensionality of 300 components. The BiLSTMs modules $\text{LSTM}_{1A}, \text{LSTM}_{1B}, \text{LSTM}_{2A}$ and LSTM_{2B} each comprise one hidden layer with 256 components, whereas the joint LSTM_{AB} comprises one layer with 512 components. We train for 5 epochs with a batch size of 32 and a learning rate of 10^{-5}. The model is trained with a dropout of 0.2 for regularization. No dropout is applied to any BiLSTM layers or the self-attention layer.

For training, development, and testing, we split our data according to the existing partition given in *wikiHowToImprove*.[8] The resulting split consists of 30,044 sentence revision pairs in the training set, 6,237 pairs in the test set, and 5,334 pairs in the validation set.

4.2 Results and Discussion

Table 3 shows the results of the pairwise ranking task. We find that our proposed model with BERT embeddings is the most accurate model for this task by a margin of about 7%. We compare our results against the baseline provided by Anthonio et al. (2020), which also makes use of ranking and a BiLSTM architecture. In contrast to our model, their baseline is a simple BiLSTM-Attention classification model using FastText embeddings. It does not use an intermediate joint representation to compare representations of two versions of an instruction. The baseline model has the advantage of being trained on individual sentences, but the increase in model accuracy for training sentence pairs by sharing context highlights the efficacy of the training regime.

Their model provides an accuracy of about 64.08% when trained and evaluated on the filtered corpus. Our model with FastText embeddings achieves an accuracy of 71.16% (+7.08%), which shows the relative importance of the joint representation.

Discussion We find that version pairs that involve a subframe relation are the easiest to distinguish across our model using both FastText and BERT, while pairs involving the usage relation are most often confused. The model using BERT embeddings performs better than the FastText-based

Model Description	Dataset	Accuracy
Anthonio et al. (2020)	Entire	74.50%
Anthonio et al. (2020)	Filtered	64.08%
Our Model + FastText	Filtered	71.16%
Our Model + BERT	Filtered	**78.40%**

Table 3: Results of the pairwise ranking task, on the full wikiHowToImprove dataset (Entire) and our subset of instructional sentences (Filtered).

Frame Relation (#errors / total)	Sentence Pair
Usage (503 / 1,965)	**Make** a comic in Flash **Create** a comic in Flash
Inheritance (352 / 1,793)	**Check** the "made in" label **Inspect** the "made in" label
Subframe (137 / 926)	**Let** your hair dry **Allow** your hair to dry
Other (160 / 443)	Next, **try** to sneak out... Next, **attempt** to sneak out...

Table 4: Some examples of sentences which our BERT-based classifier could not distinguish between the original (top) and revised (bottom) versions. We find that confusable verbs (marked in bold) are mostly synonymous. The error and total counts from the validation set are provided in parenthesis for each relation type.

[7] We will make the code available upon publication.

[8] https://github.com/irshadbhat/wikiHowToImprove

model on revisions that do not involve any frame-to-frame relations according to FrameNet (referred to as 'other' in Table 2).

In Table 4, we provide examples where the model failed using both FastText and BERT. We observe that the models fail to correctly distinguish between sentences when the main verbs are synonymous. The embeddings of the most commonly confused verb pairs, which include ⟨allow, permit⟩, ⟨choose, decide⟩ and ⟨create, make⟩, have a cosine similarity of 0.8 or higher, while the average cosine similarity between the representation of verb pairs is 0.47. This insight shows that embeddings by themselves might be insufficient for this classification task. In future work, we will explore additional features such as indicator features derived from the discourse context (e.g., the position of a sentence) and from the FrameNet resource (e.g., properties of the frames evoked in a sentence).

5 Conclusion

In this paper, we extracted a corpus of clarifications of instructions from the *wikiHowToImprove* corpus. We described a methodology for extracting version pairs of a sentence that are both instructional. We then identified cases in which a revision has clarified a vague instruction by analyzing the relationship between the frames evoked by the 'original' verb and the 'revised' verb.

In our experiments, we adopted a simple pairwise ranking task, in the same vein as performed by Anthonio et al. (2020) on the entire *wikiHowToImprove* dataset. We extended a simple BiLSTM architecture with a joint component and explored different embeddings methods, observing that both modifications lead to improvements over baselines presented in previous work.

We hope that our methodology of extracting linguistically interesting cases of revisions from a noisy dataset can be extended to more phenomena and other corpora in future work. This direction has the potential of paving the way for developing automated revision and editing methods beyond typo, style, and grammar correction.

Acknowledgements

The research presented in this paper was funded by the DFG Emmy Noether program (RO 4848/2-1).

References

Panos Alexopoulos and John Pavlopoulos. 2014. A vague sense classifier for detecting vague definitions in ontologies. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 33–37.

Talita Anthonio, Irshad Bhat, and Michael Roth. 2020. wikiHowToImprove: A resource and analyses on edits in instructional texts. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 5721–5729.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The Berkeley FrameNet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90.

Teresa Botschen, Iryna Gurevych, Jan-Christoph Klie, Hatem Mousselly-Sergieh, and Stefan Roth. 2018. Multimodal frame identification with multilingual evaluation. 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 1481–1491, New Orleans, Louisiana. Association for Computational Linguistics.

David DeVault and Matthew Stone. 2004. Interpreting vague utterances in context. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 1247–1253.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Oliver Ferschke, Johannes Daxenberger, and Iryna Gurevych. 2013. A survey of NLP methods and resources for analyzing the collaborative writing process in Wikipedia. In *The People's Web Meets NLP*, pages 121–160. Springer.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. The INCEpTION platform: Machine-assisted and knowledge-oriented interactive annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 5–9. Association for Computational Linguistics.

Logan Lebanoff and Fei Liu. 2018. Automatic detection of vague words and sentences in privacy policies. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3508–3517.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5057–5068.

André Markard and Jan-Christoph Klie. 2020. FrameNet Tools: A python library to work with FrameNet.

Timothy Meo, Brian McMahan, and Matthew Stone. 2014. Generating and resolving vague color references. In *Proceedings of the 18th Workshop on the Semantics and Pragmatics of Dialogue (SemDial)*, pages 107–115.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Stephen Roller, Douwe Kiela, and Maximilian Nickel. 2018. Hearst patterns revisited: Automatic hypernym detection from large text corpora. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 358–363.

Nathan Schneider and Chuck Wooters. 2017. The NLTK FrameNet API: Designing for discoverability with a rich linguistic resource. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 1–6.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2389–2398.

Rion Snow, Daniel Jurafsky, and Andrew Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. *Advances in neural information processing systems*, 17:1297–1304.

Yongchuan Tang. 2008. A collective decision model involving vague concepts and linguistic expressions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(2):421–428.

David Tuggy. 1993. Ambiguity, polysemy, and vagueness. *Cognitive linguistics*, 4(3):273–290.

Cristina Vertan. 2019. Modelling linguistic vagueness and uncertainty in historical texts. In *Proceedings of the Workshop on Language Technology for Digital Historical Archives*, pages 34–38.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Y Ng, and Dan Jurafsky. 2018. Noising and denoising natural language: Diverse backtranslation for grammar correction. 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 619–628.

Diyi Yang, Aaron Halfaker, Robert Kraut, and Eduard Hovy. 2016. Edit categories and editor role identification in Wikipedia. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1295–1299.

Diyi Yang, Aaron Halfaker, Robert Kraut, and Eduard Hovy. 2017. Identifying semantic edit intentions from revisions in Wikipedia. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2000–2010.