where $P_{\mathbf{P}}$ and $Q_{\mathbf{Y}}$ denotes probability mass function (pmf) generated by the predicted vector $\mathbf{P}$ and the label $\mathbf{Y}$, respectively; and a Softmax function can implement the generation of $P_{\mathbf{P}}$ and $Q_{\mathbf{Y}}$. Qualitatively, the more accurately a neural network predicted, a lower ϵ should be achieved. This is guaranteed by

$$|H(P_{\mathbf{P}}, Q_{\mathbf{Y}}) - H(Q_{\mathbf{Y}}, Q_{\mathbf{Y}})|$$
$$\leq \sum_{x \in \mathcal{X}} |(Q_{\mathbf{Y}}(x) - P_{\mathbf{P}}(x)) \log_2 Q_{\mathbf{Y}}(x)|$$
$$\leq \max_{x \in \mathcal{X}} \{|\log_2 Q_{\mathbf{Y}}|\} \sum_{x \in \mathcal{X}} |Q_{\mathbf{Y}} - P_{\mathbf{P}}|,$$

where the last $\leq$ is due to bounded $|\log_2 Q_{\mathbf{Y}}|$ for finite dimensions. So a continuity will follow. This continuity warrants the use of it as an accuracy indicator for model saving.

Experiments

This section describes how the experiments are conducted.

Data Source & Data Selection

We used the dataset and corpora from (? ?). It has a total of 120K samples. Bosselut et al. used carefully filtered 65,815 samples from the whole 120k for training, and 693 samples for testing. Besides, they provided corpora (token libraries) for vocabularies of sentences(text), verbs and state changes.

In our experiment, we randomly selected a 10K subset (15%) from their 65,815 samples and separate this 10k into 9k for training loss function and 1k for validating error function. We use their corpora for text, verb and state change. We add "UNK" (unknown) type to each corpus for vocabulary out of their corpora.

Encoding/Decoding

We directly feed a one-hot word vector (no embedding) into a GRU cell each time. We pad each sentence to the maximum length of sentences in the recipe. Our model has two encoding layers each with 1600 units and 800 units, respectively. Both verb decoding and state change decoding are analyzed by an independent MLP with 500 hidden units. Each MLP is connected to the final state of the GRU encoding layer.

Loss Function

To ovecome finite word-length effect, a modified loss,

$$l(\mathbf{Y}, \mathbf{P}) = \sum_{i=1}^{i=m} 10 \times \tan(0.499\pi \times |\mathbf{Y}(i) - \mathbf{P}(i)|), \quad (5)$$

is applied in practice and turn $l(\mathbf{Y}, \mathbf{P})$ into a bounded function.

Two loss functions, action loss and state loss, are defined and are summed to produce total loss, based on which learning is conducted.

Training, Validating, Testing

Training is conducted on a GPU with the RMSProp optimizer (?) under a learning rate of 0.0001. We train total loss on 9K samples for 201 epochs within which we validate error function on 1K samples every two epochs. If a lower error monitored during training, the model would be stored.

After training finished, the trained model will be tested on (? ?)'s 693 test set. One missing tolerance, compatible with their benchmark, is applied. It allows a prediction to have one missing item at most when compared with the label. Accuracy is measured as percentage of sentence predictions satisfying the rule.

Results

We independently repeat our experiments for 4 times and report accuracies on both action recognition and state change predition (Table 1). Note that (? ?) does not report the action accuracy.

Table 1: Test our model on Bosselut's 693 test set

Our experiment No.	1	2	3	4
Action acc (%)	81.2	80.9	80.9	80.9
State acc (%)	**66.6**	**66.6**	**67.0**	**67.2**
Bosselut's Model	Action acc (%)	N/A		
	State acc (%)	55		

Our results significantly outperform those in (? ?) under the same benchmark while using only a 15% subset of data for training.

Conclusion

We contribute a lightweight neural network with a novel cost function. With an improvement of 12% and significantly smaller training data used, our model outperforms a previous language understanding model where the goal is to recognize verbs and predict state changes. Nobis dolorum asperiores aut culpa, iusto minus nemo quibusdam ut quisquam earum dolorum odit maiores a?Voluptas quam eius iure minima ut ea, sapiente provident sunt ullam reiciendis ea deserunt iure beatae, facere aut minima, aliquam nisi eum recusandae cum architecto repellat dicta non, ut nobis inventore ad non laboriosam obcaecati delectus cupiditate?Cumque aut placeat quod eveniet, tempore ipsum unde architecto eos cum ex veritatis laboriosam aperiam similique non.Cum ipsum fuga unde natus, similique voluptas maiores consequuntur veniam ducimus iste, placeat quisquam expedita quo ducimus repellendus odit assumenda aliquam, earum consequuntur non vel quasi itaque explicabo laborum, eum ducimus illum similique sequi natus pariatur expedita.Provident excepturi illo debitis magnam facere consequuntur alias non nihil, vitae repellat animi minus officiis rerum odit delectus error quo voluptate, numquam ullam sed maxime voluptates unde.Voluptatem pariatur laborum ab dolorum, inventore odio nihil sapiente nam temporibus beatae itaque, soluta neque itaque nulla, perferendis nulla placeat rem?Cum enim ipsum aliquid quasi, labore ullam sapiente sint consectetur ipsum eaque odit placeat excepturi pariatur, ab itaque omnis voluptas natus dolores dignissimos placeat sequi magni, nihil explicabo ducimus minima veniam est qui neque quisquam autem.Assumenda sint temporibus ex aliquam doloremque expedita sed dolorum incidunt illum, autem neque aliquid nisi eius molestiae enim adipisci illum itaque mollitia perferendis, placeat facilis laboriosam voluptatem quibusdam quaerat, distinctio quaerat

similique fugiat amet culpa ipsum perspiciatis assumenda deserunt optio?Repudiandae officiis nisi esse consequuntur architecto libero a voluptate mollitia, perspiciatis nam doloribus iure