Crossword: Estimating Unknown Embeddings using Cross Attention and Alignment Strategies

Anonymous ACL submission

Abstract

Word embedding methods like word2vec and GloVe have been shown to learn strong representations of words. However, these methods only learn representations for words in the training corpus. This is problematic, as models using these representations need ways to handle unknown and new words, known as out-of-vocabulary (OOV) words. As a result, there have been multiple attempts to learn OOV word representations in a similar fashion to how humans learn new words, using surrounding words (“context clues") and word roots/subwords. However, most current approaches suffer from two problems. First, these models calculate context clue estimates and subword estimates separately and then combine them shallowly for a final estimate, therefore ignoring potentially important information each type can learn from the other. Secondly, although subword embeddings are trained to estimate word vectors, we find these embeddings don't occupy the same space as word embeddings. Current models do not take this into account, and do not align the spaces before combining them. In response to this, we propose *Crossword*, a transformer based OOV estimation model that combines context and subwords at the attention level, allowing each type to influence the other for a stronger final estimate. *Crossword* successfully combines these different sources of information using cross attention, along with strategies to align subword and context spaces.

1 Introduction

Word embeddings are very useful in natural language processing tasks. Methods like word2vec (Mikolov et al., 2013a,b) and GloVe (Pennington et al., 2014) train strong semantic representations of words using co-occurrence statistics on a large text corpus, and have been shown to be effective at semantically representing text data. However, one weakness of these methods is that they only learn representations for words that exist in the training corpus, and therefore have no representations on unknown terms, known as out-of-vocabulary (OOV) words. These terms can be new words or rare words, both of which could be very relevant to the downstream task; therefore, learning representations for OOV words is an important endeavour.

Contextualized embeddings like BERT (Devlin et al., 2018) also suffer from weak performance on rare and unknown words, despite being able to build a contextualized representation of them (Schick and Schütze, 2020). As such, the OOV problem is relevant in contextualized embeddings as well. In this work, we focus on static embeddings, as they are still very much in use for low-resource settings (e.g., data-scarce languages or domains) as well as for deploying models on small-compute devices. As a result, more static embeddings exist for more languages and domains than contextualized equivalents. For example, static embedding fastText (Bojanowski et al., 2017) covers 294 languages while multilingual BERT or XLM-R (Conneau et al., 2020) only cover 100 to 110 languages. Beyond high-resource languages, the OOV problem is especially relevant, making estimation of the representations important. Therefore, this work focuses on static embeddings, leaving OOV estimation of contextualized representations for future work.

Previous attempts mimic strategies used by humans to learn new words. Some methods (Horn, 2017; Lazaridou et al., 2017; Herbelot and Baroni, 2017; Khodak et al., 2018) use the surrounding context words an OOV word is found in, known as context clues. Other methods (Bojanowski et al., 2017; Pinter et al., 2017; Fukuda et al., 2020) use the word roots/subwords of the OOV word. The most successful attempts (Hu et al., 2019; Schick and Schütze, 2019a,b; Patel and Domeniconi, 2020) look at both context and subwords together, and combine them for a final OOV estimate.

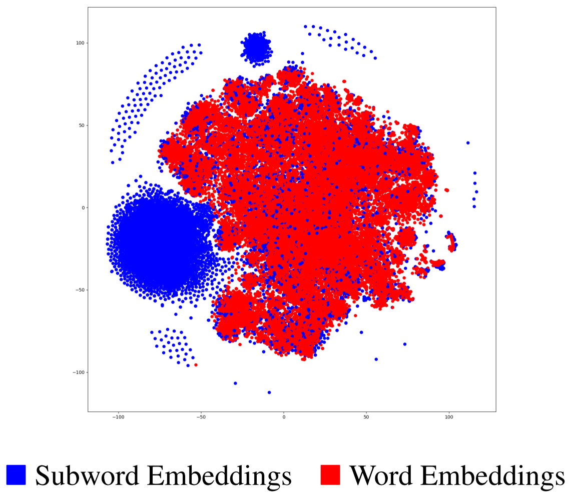


Figure 1: Subword and word embeddings clearly occupy distinct spaces (visualization with t-SNE (Van der Maaten and Hinton, 2008) over learned subword and pretrained word embeddings.)

However, current approaches that combine subwords and context do so in a shallow fashion. They usually calculate a subword estimate and context estimate separately and combine them very late in the model. Because subwords and context are combined late in the process, each estimate is not influenced by the other type of data. These approaches are missing a key advantage of combining these different types of data in order to enhance the estimate of each. For example, if we were trying to estimate an embedding for the word `octopus`, a context sentence of "An octopus has eight tentacles" could encourage a model to focus more on the word root of `oct`, as `eight` and `oct` are semantically related to each other. In this case, the context sentences can potentially encourage a stronger subword estimate. In addition, although subword representations in these approaches are trained to estimate the existing word embeddings, the two do not have the same distribution. This is shown in Figure 1, where the word embeddings are compared to subword embeddings trained to estimate them. This can weaken the combination of subword and context estimates, along with attention score calculations, as lack of alignment weakens interactions between the two types of embeddings.

This work introduces *Crossword*, a deep neural network attention model that combines subwords and context information in the attention layers (Vaswani et al., 2017) to estimate OOV words. *Crossword* uses attention mechanisms to allow each type to influence the representation of the other. It achieves this by treating the OOV estimation problem as a multimodal problem (the two modes being subwords and context), using cross attention (Bahdanau et al., 2015) to combine information from both modes. *Crossword* is shown in Figure 2, and discussed in detail in Section 3.

Crossword is a transformer based model that combines subwords and context using attention to estimate strong representations for OOV words. We make the following contributions: First, *Crossword* uses cross attention to combine subwords and context early, and improve both types' role in the final estimate. Second, we demonstrate that although subword embeddings are learned based on estimating word embeddings, they occupy different spaces, a fact that weakens cross attention calculations, and the combination of the two information types in general. We show that this is an issue and that it leads to poor alignment in the attention calculations, and between the subword and context estimates (v_{sub} and v_{ctx} in Figure 2, respectively) before their final sum. We apply alignment strategies to address this issue in *Crossword* at these two steps. Finally, we show that *Crossword* achieves state-of-the-art performance in OOV estimation, outperforming other combined subword and context approaches.

2 Background and Related Work

We now focus on relevant attention mechanisms and previous approaches to the OOV problem.

2.1 Attention

Attention mechanisms (Bahdanau et al., 2015; Sutskever et al., 2014; Vaswani et al., 2017) are an effective tool in NLP. The transformer (Vaswani et al., 2017) layers calculate attention scores with query and key representations of input, and uses these to weigh value representations. We denote self attention in the following way:

$$X_2 = \text{encoder}(X_1, X_1, X_1)$$

where the inputs refer to which group of vectors to apply the query, key, and value transformations (each input is the same in self attention). Attention mechanisms can also be used to compare one group of inputs to another, known as cross attention (Bahdanau et al., 2015). This is used to combine information from different types of inputs, making it useful in multimodal problems like (Qian et al., 2021), (Duan et al., 2020) and (Tsai et al., 2019).

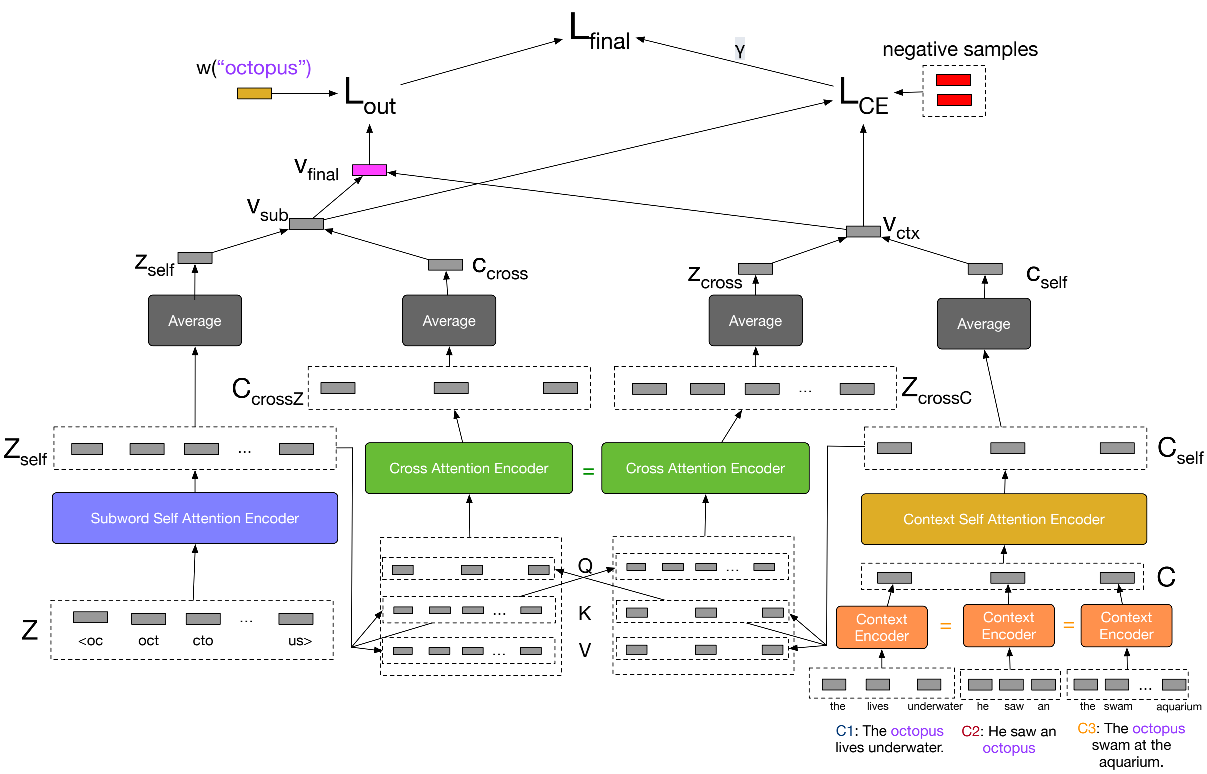


Figure 2: Crossword Architecture (best viewed in color): the estimate of "octopus" is denoted in **purple**, which is then compared to the real embedding in **gold**, while the subword estimate and context estimate are compared with each other and with negative samples in **red**.

Cross attention compares two different sequences (e.g., X_1 and Y_1), and can be represented with:

$$X_2 = \text{encoder}(X_1, Y_1, Y_1)$$

For more details on attention, see Appendix A.

2.2 OOV Estimation

As embeddings trained by word2vec (Mikolov et al., 2013a,b) and GloVe (Pennington et al., 2014) are missing OOV representations, estimating the representation of OOV words is an important endeavour. Some OOV strategies use subwords of the OOV word to estimate OOV embeddings (Bojanowski et al., 2017; Kim et al., 2018; Zhao et al., 2018; Pinter et al., 2017; Fukuda et al., 2020) while other methods use the OOV word's context (Lazaridou et al., 2017; Horn, 2017; Herbelot and Baroni, 2017; Arora et al., 2017; Mu and Viswanath, 2018; Khodak et al., 2018). However, more recent attempts combine both subwords and context approaches. Schick and Schütze (2019b) propose the Form-Context model, which estimates OOV embeddings by combining the sum of n-gram embeddings (learned by the model) with the sum of word embeddings in the contexts multiplied by a weight matrix (also learned by the model). This model has been extended to the Attentive Mimicking model (Schick and Schütze, 2019a), which adds an attention mechanism to the context calculations. A second combined approach is the hierarchical context encoder, known as HiCE (Hu et al., 2019). HiCE is a transformer based model that leverages the hierarchical structure of contexts. It uses a transformer encoder to encode each context sentence into a sentence embedding, and then uses another transformer encoder to combine each sentence embedding into a full context embedding. It estimates subword information using a character CNN, and then combines each piece into a final OOV embedding. HiCE also adapts its model to the OOV word's corpus using Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017). Another approach, Estimator Vectors (Patel and Domeniconi, 2020), trains its own word embeddings, along with subword and context embeddings for OOV estimation. While these approaches create strong estimates for OOV words, they have some weaknesses. They treat subwords and context separately, and combine them in a shallow fashion late in the model. We hypothesize that both types of information can influence the other, and therefore should be combined and interact with each other earlier in the model, something none of these methods do. Additionally, they do not align the subword and word embedding spaces, leading to weaker combinations of the two types of estimates.

Due to the weaknesses outlined above, we propose *Crossword*, a model that uses cross attention to allow individual subwords and contexts to influence each other early in the model, leading to stronger OOV estimates.

3 *Crossword*

In this section, we describe *Crossword* in detail. First, we start with motivation, then discuss architecture, and finally discuss and address alignment issues between subwords and contexts.

3.1 Motivation

As mentioned earlier, a weakness of current OOV esimation models is that they only shallowly combine subwords and context clues. We posit that this is missing out on potential information that can be used for better estimates, especially using attention. Subwords can help improve context estimates, and vice versa. For example, if estimating the word `lawyer`, with two contexts: "He wanted to be a famous lawyer or doctor" and "The lawyer read many legal documents in preparation for the court case", when trying to decide which context to emphasize more, the subwords can assist with this decision. The subword `law` in `lawyer` semantically matches the second context (with words like `legal`, `court`, and `case`), which can indicate that the second context should be focused on more.

This influence goes in the other direction as well; context can help decide which subwords to emphasize in the estimate. For example, the subword `ice` can be found in the words `iceberg` and `nice`. When estimating the meaning of these words, we may use the subword `ice` to help guess. However, in `iceberg` `ice` is extremely informative and should be weighed heavily in the estimate, while it is probably not an informative subword for `nice`. We suggest that context can help make the decision on which subwords to emphasize. `Iceberg` is likely to occur in context with words like `cold/snow`, which in turn will emphasize the `ice` subword.

This suggests early interaction between subwords and contexts is useful, and *Crossword* uses cross attention to combine both types of information, as discussed in detail in Section 3.4. However, as shown in Figure 1, the subword and word embeddings are not aligned, despite the fact that the subword embeddings are trained to estimate word embeddings. This alignment issue continues before the attention calculations and final combination of subwords and context estimates, leading to weaker attention interactions and combinations. In an effort to combat this, *Crossword* proposes alignment strategies. The attention and end alignment issues are discussed in Sections 3.5 and 3.6, respectively.

3.2 Pretraining Subword Representations

First, *Crossword* learns subword representations for the current word embeddings. We learn embeddings for character n-grams of each vocabulary word, in a similar fashion to Bojanowski et al. (2017) and Zhao et al. (2018), using the following formulation:

$$sub_{w_t} = \frac{1}{|G_{w_t}|} \sum_{g \in G_{w_t}} z_g$$

where G_{w_t} is the character n-grams (the subwords) of the word w_t, and z is the embedding of the subwords. Subword representations z are learned by maximizing the cosine similarity between sub_{w_t} and the corresponding word embedding v_{w_t}. Once these subword representations are trained, they are used in the main *Crossword* model. An OOV word is broken down into its character n-grams, which are then converted to the corresponding subword embeddings Z.

3.3 Context Encoder

For each context sentence, *Crossword* creates a representation for use later in the model. It achieves this using a context encoder similar to the one used in HiCE (Hu et al., 2019). For word w at position t in a context, the input representation q is calculated with its corresponding word embedding and a position embedding:

$$q_{w_t} = a_t \times v_{w_t} + p_t$$

with a_t a learned position weight, v_{w_t} the word embedding, and p_t a sinusoidal position encoding (Vaswani et al., 2017). These input embeddings for context j (denoted context words Q_j) are then inputted into a transformer encoder:

$$Q'_j = \text{encoder}(Q_j, Q_j, Q_j)$$

which is then averaged for a final context representation c_j:

$$c_j = \frac{1}{|Q'_j|} \sum_{q \in Q'_j} q$$

where $|Q'_j|$ is the number of context words in context j. These representations make up the context embeddings C.

3.4 Crossword Main Architecture

Crossword uses attention mechanisms on subwords, contexts, and their combination to calculate an estimate of an OOV word. Our architecture uses transformer encoder multi-head attention layers, and its cross attention is inspired by the architecture used in (Qian et al., 2021), a multimodal model used for combining image and text information. Given an OOV word and a the list of contexts it occurs in, *Crossword* calculates the OOV word embedding. First, it breaks up the OOV word into character n-grams, whose embeddings are used for the subword input (these embeddings are pretrained earlier, see Section 3.2). For the list of contexts, the context representations C are calculated using the architecture described in Section 3.3.

First, each information type is encoded through their own multi-head self attention layers:

$$Z_{self} = \text{encoder}(Z, Z, Z)$$
$$C_{self} = \text{encoder}(C, C, C)$$

Then, the self attention encodings are inputted through another set of multi-head attention layers, this time using cross attention. Two estimates are created, context estimates built out of subword embeddings as values:

$$C_{crossZ} = \text{encoder}(C_{self}, Z_{self}, Z_{self}) \quad (1)$$

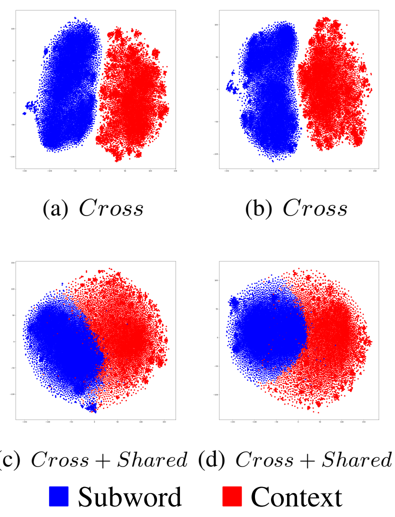
and subword estimates build out of context embeddings as values:

$$Z_{crossC} = \text{encoder}(Z_{self}, C_{self}, C_{self}) \quad (2)$$

Each group of encodings is averaged into a final representation for each attention type, creating four encodings: z_{self}, c_{self}, c_{cross}, and z_{cross}. We then combine each information type's self and cross attention for a final estimate of each type. This is done using a gated approach, similar to the one used in the Form Context and Attentive Mimicking Models (Schick and Schütze, 2019b,a):

$$v_{ctx} = \alpha_c \times c_{self} + (1 - \alpha_c) \times z_{cross}$$
$$v_{sub} = \alpha_s \times z_{self} + (1 - \alpha_s) \times c_{cross}$$

$$v_{final} = \alpha_f \times v_{sub} + (1 - \alpha_f) \times v_{ctx} \quad (3)$$



(a) $Cross$ (b) $Cross$

(c) $Cross + Shared$ (d) $Cross + Shared$

Figure 3: t-SNE plots of queries and keys in attention head 0 for C_{crossZ} (a and c) and Z_{crossC} (b and d), sampled from the validation set. In the $Cross$ model the embeddings do not align, while in $Cross+Shared$ they are closer and have some overlap. For all attention heads, refer to Appendix C.

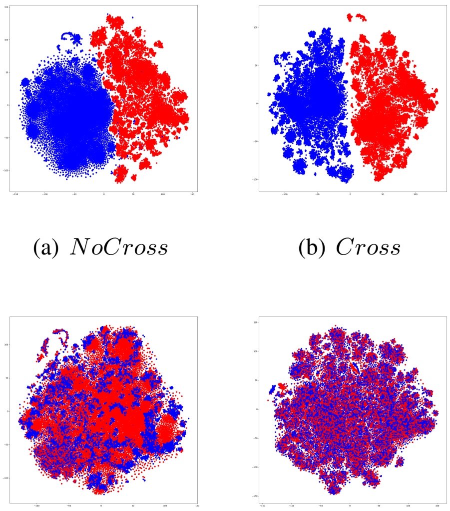
where $\alpha = \sigma(w^T[x_1, x_2] + b)$, with x_1 and x_2 as the terms being combined in the weighted sum, and σ as the sigmoid function. Equation (3) calculates v_{final} , which is our OOV estimate. $Crossword$ is trained using negative cosine similarity between the OOV estimate v_{final} and the real corresponding word embedding v_{label} as its loss function:

$$L_{out} = -\cos(v_{final}, v_{label})$$

3.5 Shared Cross Attention

Cross attention combines different information types by computing attention scores of each element of one type compared to the other type (in our case, subwords and contexts) using dot product as a similarity metric, and applying those scores to weigh each input. However, although the subwords are trained to estimate word embeddings, these embeddings occupy different spaces, an issue that continues at the attention layer. The difference in embeddings leads to different spaces between the query and key vectors, as shown in the cross attention model (denoted as $Cross$) in Figure 3. This misalignment can lead to weaker attention score calculations, as attention scores are based on similarity between specific queries and keys.

To improve alignment at the attention level, *Crossword* uses the same weights for both cross attention modules, meaning the encoders used in Eqs. (1) and (2) are the same. This means that for each query, key, and value calculation in the en-



(a) $NoCross$ (b) $Cross$

(c) $Cross + Shared$ (d) $Cross + Shared + CE$

Subword Context

Figure 4: t-SNE plots of subword and context estimates before the final combination, sampled from the validation set. $NoCross$ and $Cross$ have unaligned spaces; $Cross+Shared$ is more aligned but still has clusters of each type. $Cross + Shared + CE$ is the most aligned.

coder, the cross attention has to work with both the context inputs and subword inputs. For example, the query transformation has to transform contexts in Eq. (1) and subwords in Eq. (2), to match the corresponding key transformations of subwords and context respectively. This encourages both representations to be more similar before they are used in cross attention calculations, which in turn improves the attention estimates. As shown in Figure 3, the $Cross + Shared$ model has subword and context representations that are closer and with more overlap than just the cross attention model.

3.6 Contrastive End Loss

In addition to the attention level, we demonstrate that the final combination of the subword OOV estimate and context OOV estimate suffers from misalignment. *Crossword* calculates a subword estimate and a context estimate, and then combine them afterwards. However, this combination is not very effective if the subword and context estimates are not in the same space. As shown in Figure 4, in $Cross$ and an equivalent model which replaces the cross attention with self attention (denoted as $NoCross$), these estimates are misaligned based on their type. Additionally, while $Cross+Shared$ has a much stronger alignment between subwords and context, the subword representations still are somewhat grouped together. In an effort to join the spaces even more and create a stronger combination of subword and context estimate, we use a contrastive loss function to push the representations closer together. This loss is calculated using triplet loss (Faghri et al., 2018; Wang et al., 2014), which rewards the similarity of a target pair (the subword estimate and the context estimate) while discouraging similarity with each estimate and a negative sample, taken from a different sample in the same batch during training. Two contrastive losses are used, one with a negative subword sample and one with a negative context example:

$$L_{CE1} = \max(\cos(\widehat{v}_{sub}, v_{ctx}) - \cos(v_{sub}, v_{ctx}) + m, 0)$$

$$L_{CE2} = \max(\cos(v_{sub}, \widehat{v}_{ctx}) - \cos(v_{sub}, v_{ctx}) + m, 0)$$

$$L_{CE} = L_{CE1} + L_{CE2}$$

where $\widehat{v}_{sub}$ and $\widehat{v}_{ctx}$ are negative samples, and m is a margin term hyperparameter. The contrastive losses are then combined with our main loss for a final loss function:

$$L_{final} = L_{out} + \gamma L_{CE}$$

where γ is a hyperparameter. As shown in Figure 4, adding this contrastive loss (denoted $Cross + Shared + CE$) successfully merges the subword and context spaces before the final combination.

4 Experiments

We now describe how *Crossword* is trained and evaluated, along with how its results compare to other OOV methods.

4.1 Training Corpus and Word Embeddings

The goal of *Crossword* is to estimate representations for OOV words given existing word embeddings. For the gold standard word embeddings, we use the embeddings provided by Herbelot and Baroni (Herbelot and Baroni, 2017), as done in previous OOV models like (Schick and Schütze, 2019b) and (Hu et al., 2019). For training models, contexts are taken from the Westbury Wikipedia Corpus (WWC) (Shaoul, 2010). We use the version from (Khodak et al., 2018) with certain words filtered out for the Contextualized Rare Word Task (see Section 4.3). Additionally, as Van Hautte et al. (2019) note, current OOV evaluation tasks benefit from words of the same stem in the training set, even if the original word is filtered out. To combat this, we filter out all words that share a stem with words in the Contextualized Rare Words task and Chimera task, similar to the approach in (Van Hautte et al., 2019).[1] The filtered WWC was preprocessed using the preprocessing script provided by Schick and Schütze (2019b), creating a set of words to learn along with context sentences those words appear in. All models are trained using this dataset.

4.2 Baselines and Hyperparameters

We now demonstrate the effectiveness of *Crossword*.[2] We compare it to Attentive Mimicking[3] (AM) model and HiCE[4], as they are OOV models that use both subwords and context on existing word embeddings. Two versions of HiCE are examined; the default with a 2 layer context aggregator, and a version with 8 layers to be more comparable to *Crossword* (which uses 4 layers in each self and cross encoder). Also, we do not use MAML in the HiCE experiments, in order to focus on how the architecture adapts to multiple OOV tasks. The data set and vocab is split into a training and validation set for hyperparameter tuning. Data preprocessing, hyperparameter tuning and implementation detail are discussed in further detail in Appendix B.

Ten final trials of each model are trained and then each model is evaluated on various OOV tasks. The results are tested for statistical significance using a one-way ANOVA with a post-hoc Tukey HSD test with a p-value threshold equal to 0.05. In Table 1 the best score is presented in bold, along with any scores that are not significantly different from the best.

4.3 Tasks

We now evaluate *Crossword* on various OOV tasks. We focus on OOV tasks in English, matching previous work. As *Crossword* mixes both subwords and contexts, we select OOV tasks with high quality subwords: the Contextualized Rare Word Task in Section 4.3.1 and a subword-adapted version of the Chimera Task in Section 4.3.2.

4.3.1 Contextualized Rare Word Task

The Contextualized Rare Word task (CRW; Khodak et al., 2018) is built off the Rare Word data set (Luong et al., 2013), which is a list of rare words

[1] Note that the Chimera Task words filtered are based on the words used to build the chimeras, see Section 4.3.2 for more details.

[2] Implementation will be available at `AnonymizedURL`

[3] https://github.com/timoschick/form-context-model

[4] https://github.com/acbull/HiCE

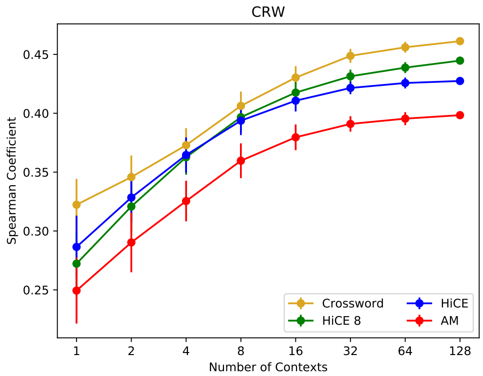


Figure 5: CRW Task - *Crossword* outperforms all competitors in all context sizes, demonstrating its strength in OOV estimation.

paired with other words, along with human similarity scores. Khodak et al. (2018) added contexts to this set, allowing for OOV words to be estimated using both subwords and context. The goal is to output an OOV embedding, compare it to the other words, and evaluate the scores' correlation with human judgements. CRW has a large range of context sizes, from 1 to 128, so the quality and informativeness of the context can vary wildly. However, the words gathered for the Rare Word set have intentionally informative word roots, and therefore we expect subwords to be fairly informative.

The results of the CRW task are shown in Figure 5. *Crossword* significantly outperforms all competitors in all contexts, showing its effectiveness as an OOV estimator. This shows the strength of deeply combining subwords and context, along with aligning the spaces. We note that after 4 contexts, as the number of contexts increases, the amount by which *Crossword* outperforms competitors generally increases as well. We theorize more contexts lead to even stronger cross estimations (as there is more information to emphasize each other) in addition to the stronger context estimates.

4.3.2 Chimera Task

The Chimera Task (Lazaridou et al., 2017) creates fake words (the “chimeras”) by combining two real words, and then puts the “chimera” word in a passage made from sentences extracted from the corresponding real words. For example, the chimera `divirth` is a fake word that "occurs" in contexts built by combining passages from the words `corn` and `yam`. These passages are then semantically compared with various probe words, with similarity scores given by human judgements. The goal of this task is for a model to estimate the embedding

	L2	L4	L6
AM	0.3177	**0.3765**	0.3945
HiCE	**0.3240**	**0.3746**	**0.3973**
HiCE 8 Layer	0.3186	**0.3719**	0.3925
Crossword	**0.3289**	**0.3756**	**0.4030**

Table 1: Chimera - Correlation with human similarity scores. *Crossword* outperforms or ties other models.

of the chimera, calculate its similarity to the known probe words, and then see how well its similarity scores correlate with human given scores. The better the correlation, the closer the model is to a human judgement. The chimera task has 3 sets of passages; 2, 4, and 6 sentence size passages (called L2, L4, and L6). To fit our problem better, we make two changes to the traditional chimera task. First, since the models we are viewing combine subwords and context, we take the context from the passage as normal, but use the original words concatenated to each other for the subword information (for example, `divirth` is replaced with `cornyam`). This allows the task to have relevant subword information , unlike the original task. Secondly, we increase the size of the evaluation data by combining the chimera test sets with the chimera train sets, as the train set is not used for any tuning. This allows a bigger set to be used for evaluation. The Chimera Task results are shown in Table 1. *Crossword* either outperforms or ties with competitors in all tasks. For L2 and L6, it outperforms AM and HiCE 8 Layer, while tying (in terms of significance) with HiCE. In L4, all models tie. *Crossword* performs well in this task, with HiCE performing just as well. We suspect *Crossword* ties with HiCE (as opposed to exceeding it) in this setting due to the low number of contexts. With relatively fewer contexts, there is less information for the cross attention calculations, hence the advantage of *Crossword*'s cross attention is smaller. Fewer contexts means less cross-enhancement of the subwords, and less information for the subwords to enhance. Despite these challenges, *Crossword* still performs well and en par with other models.

4.3.3 Ablation Study

Finally, we conduct an ablation study on *Crossword*, shown in Figure 6. *Crossword* is denoted as $Cross + Shared + CE$, because it uses cross attention, shared encoders, and contrastive end loss. We remove the contrastive end loss in model $Cross + Shared$, remove the shared encoder for $Cross$, and remove the cross attention (replacing it with more self attention layers) in $NoCross$. As shown in the figure, $Cross{+}Shared{+}CE$ (*Crossword*) outperform or ties all models in all contexts. In smaller context sizes it matches $NoCross$ and $Cross$ (in significance) while outperforming $Shared$, while in higher context sizes it matches $Shared$ for best (in significance) while outperforming $NoCross$ and $Cross$. In lower context sizes, we suspect $Shared$ underperforms due to its stronger reliance on cross attention, which may be weaker with less context information. This also explains its strong performance in high context sizes. $Cross{+}Shared{+}CE$ seems to escape this weaker performance, which suggests the alignment at the end estimates (CE) makes up for this issue. Interestingly, it seems the $Cross$ also doesn't suffer from this problem, but does not perform well in later contexts. We theorize that this is due to the misalignment in $Cross$ at the cross attention layers, forcing the model to rely on its self attention layers instead, making it perform similarly to $NoCross$.

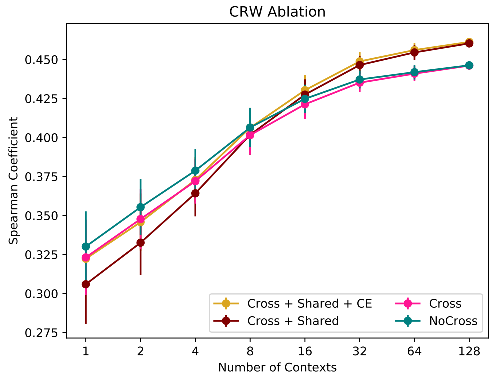


Figure 6: Ablation CRW Task - *Crossword* is the best model; removing CE continues the strong performance at high contexts but performs worse at weaker contexts; removing $Shared$ weakens performance in high number of contexts.

5 Conclusion

We propose *Crossword*, an attention based model that estimates OOV words by combining subwords and contexts in a deep manner. It achieves this using cross attention and alignment techniques to ensure a strong combination of subword and context features. We show through various experiments that this model estimates more accurate representations of OOV words. In the future we plan to extend this work by studying how well *Crossword* performs at estimating OOV embeddings in contextualized embedding models like BERT.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International conference on learning representations*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*, 5(1):135–146.

François Chollet et al. 2015. Keras. https://keras.io.

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 *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Chaoqun Duan, Lei Cui, Shuming Ma, Furu Wei, Conghui Zhu, and Tiejun Zhao. 2020. Multimodal matching transformer for live commenting. *arXiv preprint arXiv:2002.02649*.

Fartash Faghri, David J. Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2018. Vse++: Improving visual-semantic embeddings with hard negatives. In *BMVC*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org.

Nobukazu Fukuda, Naoki Yoshinaga, and Masaru Kitsuregawa. 2020. Robust backed-off estimation of out-of-vocabulary embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4827–4838.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Aurélie Herbelot and Marco Baroni. 2017. High-risk learning: acquiring new word vectors from tiny data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 304–309.

Franziska Horn. 2017. Context encoders as a simple but powerful extension of word2vec. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 10–14.

Ziniu Hu, Ting Chen, Kai-Wei Chang, and Yizhou Sun. 2019. Few-shot representation learning for out-of-vocabulary words. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4102–4112.

Mikhail Khodak, Nikunj Saunshi, Yingyu Liang, Tengyu Ma, Brandon M Stewart, and Sanjeev Arora. 2018. A la carte embedding: Cheap but effective induction of semantic feature vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22.

Yeachan Kim, Kang-Min Kim, Ji-Min Lee, and SangKeun Lee. 2018. Learning to generate word representations using subword information. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2551–2561.

Angeliki Lazaridou, Marco Marelli, and Marco Baroni. 2017. Multimodal word meaning induction from minimal exposure to natural text. *Cognitive science*, 41:677–705.

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.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *ICLR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jiaqi Mu and Pramod Viswanath. 2018. All-but-the-top: Simple and effective post-processing for word representations. In *6th International Conference on Learning Representations, ICLR 2018*.

Raj Patel and Carlotta Domeniconi. 2020. Estimator vectors: Oov word embeddings based on subword and context clue estimates. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword RNNs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 102–112.

Shengsheng Qian, Jinguang Wang, Jun Hu, Quan Fang, and Changsheng Xu. 2021. Hierarchical multimodal contextual attention network for fake news detection. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 153–162.

Timo Schick and Hinrich Schütze. 2019a. Attentive mimicking: Better word embeddings by attending to informative contexts. 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 489–494.

Timo Schick and Hinrich Schütze. 2019b. Learning semantic representations for novel words: Leveraging both form and context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6965–6973.

Timo Schick and Hinrich Schütze. 2020. Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8766–8774.

Cyrus Shaoul. 2010. The Westbury lab Wikipedia corpus. *Edmonton, AB: University of Alberta*, page 131.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Yao-Hung Hubert Tsai, Shaojie Bai, Paul Pu Liang, J. Zico Kolter, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2019. Multimodal transformer for unaligned multimodal language sequences. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Florence, Italy. Association for Computational Linguistics.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Jeroen Van Hautte, Guy Emerson, and Marek Rei. 2019. Bad form: Comparing context-based and form-based few-shot learning in distributional semantic models. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 31–39.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. 2014. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1386–1393.

Jinman Zhao, Sidharth Mudgal, and Yingyu Liang. 2018. Generalizing word embeddings using bag of subwords. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 601–606.

A Attention Details

The transformer uses an attention mechanism known as multi-headed attention. For input vectors X, an attention head calculates query vectors Q, key vectors K, and value vectors V:

$$Q = X \times W_Q$$

$$K = X \times W_K$$

$$V = X \times W_V$$

where W_Q, W_K, W_V are linear transformations learned by the model. Then, for each input, its query vector q_i in Q is paired with each key vector k_j in K to calculate attention scores:

$$a_{ij} = softmax(\frac{q_i * k_j}{\sqrt{d}})$$

where d is the dimensionality of the key vectors. Then, these attention scores are used in a weighted sum of each value vector in V to calculate the output representation of that embedding:

$$out_i = \sum a_{ij} v_j$$

In addition, the transformer attention mechanism uses multiple W_Q, W_K, W_V matrices, known as multi-headed attention. The output from each head is concatenated and multiplied by a final linear transformation W_o for a final output of the mechanism. After the attention block, each output is layer normalized (Ba et al., 2016) and then combined with the input using a residual connection (He et al., 2016). This is passed through a feed-forward neural network, which then uses another layer normalization and residual connection step. The attention block and feed-forward block combine to make the transformer's encoder layer. For self attention, the attention mechanism compares the input sequence to itself, so the encoder block is denoted in the following way:

$$X_2 = \text{encoder}(X_1, X_1, X_1)$$

where the inputs refer to which group of vectors to apply the query, key, and value transformations. Since it is self attention, these are all the same input X_1.

In addition to self attention, multi-headed attention can be used to compare one group of inputs to another group, known as cross attention (Bahdanau et al., 2015). In the transformer, cross attention uses the same structure as the self attention, but uses one group for the query vector calculation and the second group's vectors for the key and value vector calculation:

$$X_2 = \text{encoder}(X_1, Y_1, Y_1)$$

where X_1 and Y_1 are each sets of input vectors of different types.

B Implementation Details

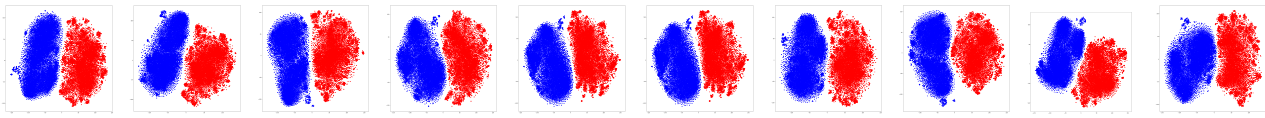
For the training and validation set, the vocabulary is split into a training set and validation set, similar to the training approach in (Hu et al., 2019). Words are grouped by stem (this avoids overly informative subwords) and a train set and validation set are built, with around 90% of groups making up the training set and around 10% making up the validation set. The subword n-grams used in AM and *Crossword* are extracted on the training set words. In an effort to reduce subword overfitting, these character n-gram models randomly drop out subword n-grams during training. All models were trained and validated on a varying number of contexts (1 to 64), as done in (Schick and Schütze, 2019a).

Crossword is implemented in Keras (Chollet et al., 2015). For AM, we use an edited version of the code presented in the author's github, adapted to work with a training and validation set. Similarly, we use the HiCE author's implementation adapted to work with the WWC training corpus. In *Crossword*, the context encoder has two layers, while the self and cross encoders have 4 layers each. In our experiments, we use two HiCE models; one with 2 layers for the context aggregator (the default), and one with 8 layers, in an effort to be more comparable to *Crossword*.

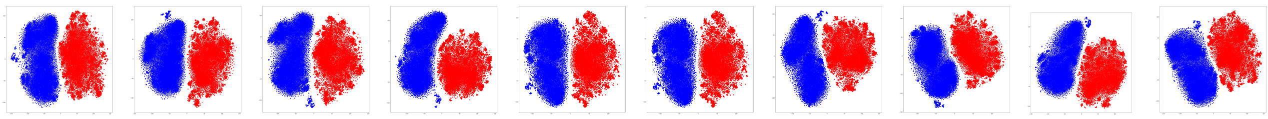
The best hyperparameters are found based on the validation loss, with the best epoch selected. First, learning rate is selected, then n-gram dropout (based on the selected learning rate). Note that HiCE does not use n-gram subwords, so n-gram was not used in the model. For *Crossword*, γ and margin m were not validated on, simply choosing .01 and 0 respectively.

C All Attention Heads for Attention Level Visualization

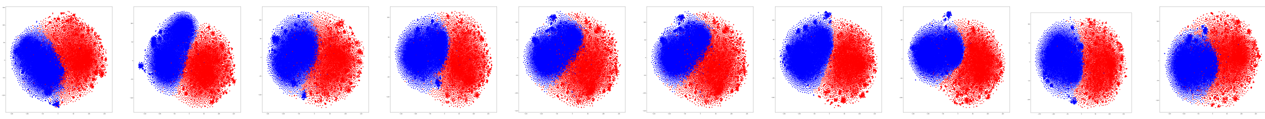
Here we extend the attention level visualization in 3 to all attention heads.



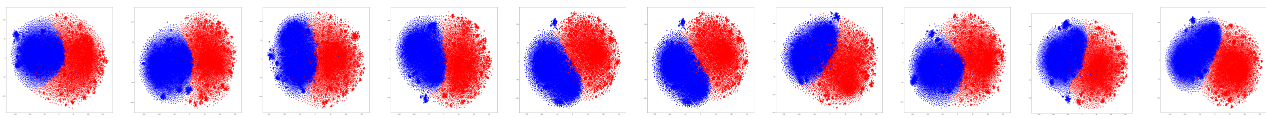
(a) $Cross\ C_{crossZ}$: Attention Heads 0 - 9



(b) $Cross\ Z_{crossC}$: Attention Heads 0 - 9



(c) $Cross + Shared\ C_{crossZ}$: Attention Heads 0 - 9



(d) $Cross + Shared\ Z_{crossC}$: Attention Heads 0 - 9

Subword Context

Figure 7: t-SNE plots all attention heads