QACNNnet - Model performance improvements through Data Augmentation

Gaetano Signorelli[1*] **Daniele Sirocchi**[1*]
[1]University of Bologna
gaetano.signorelli@studio.unibo.it
daniele.sirocchi@studio.unibo.it

Abstract

One of the biggest concerns about neural networks is that they tend to require a huge amount of training data, otherwise they typically can't operate properly. Data Augmentation is a fundamental technique for facing their data-hungry behaviour, and its goal is indeed to increase the dataset size, by creating new data through the transformation of those that are available. This work aims to assess the usefulness of the data augmentation, so as to demonstrate that it can be effectively used for increasing the performance of a neural network in the field of Question Answering. Moreover, an extensive study has been conducted to make the training procedure faster and more efficient, with the goal of reducing the costs and the time required by these kinds of models. Finally, an analysis of the obtained results summarizes the strengths and the weaknesses of these methods, and some alternatives have been proposed for possibly further improving the behaviour of the model.

1 Introduction

Data augmentation represents a primary way of increasing the size of a dataset, a characteristic that is essential for training and improving the performances of neural networks. It is important to remark that data augmentation can be applied to a wide variety of data types, like images and texts. However, depending on the type, the techniques for augmenting the data are very different. For example, a dataset of images could be augmented by performing operations such as translating, rotating and flipping these inputs; while a dataset of texts requires totally different kinds of transformations, and some of them have been explored and evaluated for the development of this work.

Particularly, data augmentation has been applied on the SQuAD [7] dataset, which is made up of samples composed of a context, a question and an answer. Moreover, this dataset has the characteristic of having the answer always contained in the context. For these reasons, the SQuAD is one of the most common datasets used for training models meant to solve the Question Answering (QA) task, whose goal is to find suitable answers to questions made by users. Eventually, the augmented version of the SQuAD dataset has been employed for retraining the QACNNnet [2], a lightweight model originally inspired by [1], and created for facing the QA task. This model is built on a series of convolutional layers and self-attention layers used to catch, respectively, local and global text relationships. Two different data augmentation phases have been deployed: *paraphrase* and *multilingual embedding*. The former is meant to write the same content, but with different words; while the latter leverages the usage of more than one language for enhancing the dataset.

In order to assess the effectiveness of a certain augmentation technique with respect to the others, a baseline network has been created and used as a reference for all the conducted tests, which are covered in the **Experiments** section. This baseline network differs from the original QACNNnet for the word embedding method used. Specifically, for supporting the *multilingual* *embedding*, the GloVe [4] word embedding has been replaced with fastText [3], which has the peculiarity of supporting several languages, while also introducing a unified and aligned vector space for all these languages (namely, similar words of different languages are identified by similar vectors). More details on this can be found in the **Multilingual embedding** subsection.

Lastly, this work suffers from a big issue: the number of experiments is large, and training the QACNNnet needs several hours. This situation has required the study and the implementation of techniques for making the neural network model lighter [5], but without losing the accuracy of the results. In particular, a very interesting Tensorflow's feature, called *mixed precision*, has been implemented, and it has allowed to heavily reduce time and costs necessary for training the whole architecture. Refer to the **Computational saving** section for a more exhaustive explanation.

2 Data augmentation

The main attempt to increase the accuracy of the discussed model has been represented, as previously pointed out, by means of data augmentation of the training dataset. Especially, being this one a text-based network, respecting specific rules (e.g., the presence of three blocks of text linked to each other by the relation context-query-answer), this step has been centred on building a parallel dataset, made up of the paraphrased sentences of the original one; with the ultimate goal of merging the two datasets into a single one, and to use it for training the network.

Paraphrasing the entire corpus has been so treated as a translation task. In particular, many advanced translation systems are based on pre-trained neural models that are able to grasp the local meaning of entire sentences, which allow them to express the same concepts in another language, without being oriented on a dictionary-based translation. Fully or partially algorithmic approaches to carry out the translation of complex texts, in fact, would not have been useful for the scope of this work, as they show the tendency to transcribe by following the behaviour of a bijective function. Modern models, on the other hand, allow to switch to another language and then translate back to the original one, getting a different text with respect to the starting one (i.e., the output is paraphrased, as requested).

The choice of the model has fallen into *Argos Translate* [9], which in turn exploits the OpenNMT's *CTranslate2* model [10]. It contains the most useful features required, as it is:

- *Powerful*: it relies on the proved strength of a transformers-based architecture to accomplish high quality translations;
- *Lightweight*: the pretrained model has been revised in order to maintain the same accuracy but sensibly accelerating the inference time and reducing the memory usage, thanks to many performance optimization techniques such as weights quantization, layers fusion, batch reordering, etc. [9];
- *Multilingual*: it currently supports 30 of the most common languages, making it possible to perform all the needed tests (see **Experiments** section);
- *Reliable*: this model, which has been stressed in many ways in order to find potential vulnerabilities, shows an almost perfect stability: it always returns a valid output, without, for instance, truncated sentences, meaningless words or long repetitions of words at the end of the translation (all of these are typical issues encountered with other models, previously taken into account);
- *Simple*: it does not necessitate any pre-processing operations.

2.1 The indexing problem

Due to the nature of the given dataset, translating the texts was not enough in order to build a second and paraphrased corpus: one of the fundamental points is that the answer must always be a part of the context, appearing there with the same number and type of characters. This point has proved to be the single disadvantage of using the discussed model: the translation of a text is context-influenced, so a different result has been achieved (in a minor number of cases) between the translated answer and the translated context, which did not include the former anymore. More formally, it was impossible to assign the attribute *start index*, namely the index of the first answer's character found in the context, and therefore invalidating the augmented (i.e., translated) sample.

Triples (i.e., context-query-answer) affected by this phenomenon have eventually been discarded, because the problem has been evaluated as intractable to solve. In other words, no practical solution has been identified to keep the translation valid, thus to force a match between answer and context. The responsibility of this fact is related to a a series of reasons, explained below.

First of all, the problem can be formulated as finding within the context a string such that it is the closest one, in terms of similarity, to the answer, and then replacing it. However, algorithmically all the trivial attempts are destined to fail, due to the wide (and hard to fully cover) range of possibilities:

- one or more words have been written differently between the answer and the context;
- one or more words have been replaced with synonyms;
- one or more words have been replaced with entire expressions (two or more words) keeping the same meaning, but increasing the number of words inside of the answer (e.g., *thanks* and *thank you*);
- as above, but using expressions that reduce the total number of words belonging to the answer;
- couples of words have been placed in reverse order, such as in case of phrasal verbs (e.g., *pick someone up* and *pick up someone*);
- a combination of two or more points listed above (e.g., *To give someone the rubber* and *To give the eraser to someone*).

The overall conclusion is that there is no stable point that can be used to build a valid algorithm, since answer and context may differ for anything: words, their order and even their count. Moreover, there is no guarantee that a selected candidate string inside of the context will correspond to the real answer, not having at disposal another algorithm to check the validity of the selection; which implies that such a dataset, created in a way that tries to maintain as many records as possible without discarding them, could not be verified and it could potentially contain artifacts where the given answer is completely wrong and misleading. In other words, there are no known algorithms (i.e., the edit distance approach) that could be used to solve (at least partially) the problem, without posing other risks.

Another infeasible, and discarded, solution has been one involving other neural architectures pre-trained on the *sentence similarity* task, which, once again, would be subject to a low reliability and a quite expensive computational cost: they would require to compare the answer with many possible n-grams within the context to find the highest similarity score.

Finally, translating the answer by enhancing it with contextual information, would run into the impossibility of discerning between the translated answer and the translated extra context.

All the stated implications and assumptions have led to the above-mentioned decision of removing, from the translated dataset, all the records affected by the examined inconsistency. This choice has implied the deletion of around 20-30% triples (at most), still allowing for a sensible and fully reliable data augmentation.

2.2 Multilingual embedding

As it will be explained later (see **Experiments** section), some tests have been conducted using a type of paraphrases that did not involve the back translation step, making the new dataset a hybrid of two or more languages. In order to let this work with the existing architecture, a *multilingual embedding* has been required. The main idea is to use a system where similar words in different languages get similar embeddings, meaning that they are close in the vector space. Conversely, different words get dissimilar embeddings, and they are more separated in the vector space.

The original model [2] was especially based upon fixed embeddings that were taken from the GloVe [11] embedding system. Keeping it as the core to get embeddings also for other languages has shown a crucial problem: each language has a specific representation that describes all its words inside of a separate vector space; thus, violating the above expressed principle.

This particular problem has been tackled in recent years, and it is still a topic under research. However, some interesting solutions have been experimented. Most of them, are based on the idea of matrix transformation: defining V_1 a target vector space (usually the one containing the embeddings in English) and V_2 a source vector space (the one related with a non-English language) containing n vectors (i.e., embeddings), the goal is to find a matrix W such that:

$$min_W = \sum_{i=1}^{n} || \; y_i - W x_i \; ||^2 \tag{1}$$

Where X_i is a the vectorized form of a word in V_2, and y_i is its closest translation in the language represented by V_1.

In other words, minimizing (1) allows to find a transformation that leads to a single vector space (V_1), in which the sought property (i.e., similar *semantic meaning* correlated with *closeness* in space and vice versa) is satisfied. It is important to highlight that the transformation matrix W must be orthogonal with respect to the source vector space: in this way, its vectors will just be translated and rotated, and their relationships will be safeguarded. Otherwise, all the internal properties (mainly *cosine similarities*) associated with its embeddings could be lost.

Different approaches and more advanced techniques have been applied to minimize (1), getting to vector projections that achieve high and reliable scores, which can be measured exactly by (1). The discovered transformation matrices are also called *alignment matrices*, and they can be used to project embeddings related to a wide variety of languages onto a single space, reaching a coherent representation of the words, despite coming from different sources. This mechanism has been chosen as a solution to have a *multilingual embedding* for the model.

Due to the high computational costs, it has been decided to not manually derive the alignment matrices for all the source languages, but to make use of the pre-aligned embeddings, available online [12], based on *fastText*'s embedding system, which represents a reasonable alternative to *GloVe* in terms of general performances. The aligned matrices used by *fastText* are based on a state-of-the-art technique, presented by *Facebook AI Research* [13], which ensures high projecting precisions, with an average score (obtained as output from the Equation (1)) of 84.7% for French, 87.1% for Spanish and 77.6% for German.

3 Experiments

There are plenty of possibilities for creating suitable experiments about data augmentation, and it has been remarkably important to find a criterion for limiting the number of these experiments, while trying to maximize as much as possible the data augmentation effectiveness.

First of all, a baseline model was created, and it differs from the original QACNNnet for the word embedding method used. More precisely, as mentioned before, the *GloVe* word embedding was exchanged in favor of *fastText*. Once the model has been updated with this new word embedding method, then the default (i.e., not augmented) SQuAD dataset has been used for training it from scratch. Afterward, the trained model has been evaluated on the SQuAD test set, and the obtained *F1* and *Exact Match (EM)* scores have been used as references for all the following experiments. This way of proceeding allows to grasp which data augmentation techniques are better than others, at least according to the considered metrics.

At this point, the experiments have been split into two macro-categories:

		AUGMENTATION			TEST SCORES	
Test Name	**Language**	**Query**	**Context**	**Answer**	**F1**	**EM**
Baseline	en	-	-	-	73.6842	62.3368
TestA1	en-fr	✓	✗	✗	71.1373	58.2024
TestB1	**en-fr**	✗	✓	✓	**74.1664**	**63.1693**
TestC1	en-fr	✓	✓	✓	72.7126	62.3746
TestB2	en-de	✗	✓	✓	74.6946	62.9328
TestF2	en-fr-de	✗	✓	✓	71.0096	57.5591

Table 1: First round of experiments, consisting in a *Monolingual paraphrases* of different parts of the input. TestB2 and TestF2 are essentially the same of TestB1 (i.e., the one providing the best scores), but involving different languages.

- *Monolingual augmentation*, where a language model has been used for translating the original SQuAD dataset from English to another language (see **Data augmentation** section), and then back translated to English.
- *Multilingual augmentation*, that essentially consisted in removing the just mentioned back translation step, so as to evaluate whether the model was able to perform better while trying to manage multiple languages.

Both categories have consisted in the usage of different additional languages: French, German and Spanish. This choice has been guided by considering the alignment quality score between the English language and the others, an aspect that is well documented in the fastText paper [13] and reported inside of the **Multilingual embedding** subsection.

Then, it has been considered the aspect related to the SQuAD dataset composition, based on samples that are essentially triplets, made up of a question, an answer and a context. These triplets have been augmented in an incremental manner, so as to assess which augmentation was actually boosting the model's performances. Specifically, for the *Monolingual augmentation*, the first experiment consisted in augmenting the query, the second in augmenting the context and the answer, and the third in augmenting all the elements in a triplet. Note that the context and the answer have been always augmented together, otherwise it would have been impossible to remain consistent with the SQuAD dataset and the QACNNnet assumption, which expect that the answer is always within the context. Finally, in order to limit the number of experiments, the tests made for *Multilingual augmentation* have been conducted by augmenting only those parts of the triplets that provided the best results in the previously made experiments.

Every experiment in each category involved the retraining of the QACNNnet from scratch for 60 epochs [2], using an augmented version of the original English SQuAD dataset, obtained by appending to it all the augmented samples. Therefore, according to the number of languages used for enhancing the dataset, the final number of samples has been roughly doubled (when using only one language for augmentation), or triplicated (when using two languages for augmentation).

Table 1 and Table 2 summarize the results of the conducted tests. In particular, Table 1 shows the results of the experimentation conducted on the *Monolingual augmentation* category, depicting the highest scores are obtained when augmenting only the context and the answer, and using the French language; while Table 2 shows the outcomes when running the model leveraging the *Multilingual augmentation*, proving that the best scores are acquired again with the French language. With respect to the baseline model, these outcomes exhibit a gain of roughly 1% and 1.7%, respectively for the *F1* and *EM* metrics.

In conclusion, from these results, it seems that the French language is the best choice when coupled with English, providing an overall good improvement compared with the baseline model. Refer to the **Results** section for further detail.

		AUGMENTATION			TEST SCORES	
Test Name	**Language**	**Query**	**Context**	**Answer**	**F1**	**EM**
TestD1	**en-fr**	✗	✓	✓	**74.6971**	**64.0681**
TestD2	en-de	✗	✓	✓	71.6729	57.6443
TestD3	en-es	✗	✓	✓	70.3135	58.7519
TestF1	en-fr-de	✗	✓	✓	71.4276	58.4484

Table 2: Second round of experiments, consisting in using the *Multilingual paraphrases*. In the first 3 experiments what changes is the second language used by the model, while TestF1 uses the combination of the most promising languages (i.e., those that provided highest scores).

4 Computational saving

Deep neural networks are very widely used, and they are often the main tool utilized for reaching the current state-of-the-art in many Artificial Intelligence tasks. However, these networks are used for coping with problems characterized by an increasing complexity, causing an increment of the required computational capacity, and this is becoming a relevant problem. Therefore, finding techniques able to reduce the computational complexity without losing accuracy is a very active area of research.

The model used for this work (i.e., QACNNnet) can be considered a lightweight model, as explained in [2]. Nevertheless, a single epoch trained on the original (i.e., not augmented) SQuAD dataset needed about 15 minutes using an *NVIDIA Tesla A100*, meaning that about 15 hours of training were necessary to carry out a full 60 epochs-long training. Moreover, knowing that with data augmentation the dataset would have had its size either almost doubled or tripled, the required time to finish the experiments (see previous section) could have reached up to 45 hours of training, essentially making the whole work extremely costly and computationally very demanding.

Rather than a problem, this has been seen as an opportunity for experimenting one of the techniques suggested by the literature [5]. Specifically, the adopted technique is meant to transform all the numbers used in computations from *float32* (4 bytes in memory) to *float16* (2 bytes), therefore drastically reducing the memory consumed by the model, but (almost) without losing accuracy. Conveniently, Tensorflow (i.e., the framework utilized for creating the network) has some specific API that can be used [6], and that should transform all the *float32* used by the model in *float16*, with just a few lines of codes. However, even though the training time for 1 epoch passed from approximately 15 minutes to less than 7, the network became affected by *numerical instability*, that ultimately resulted in a non-learning network with an *inf* loss.

For these reasons, a long phase of code debugging led to the discovery of a potential incompatibility between the implementations of Tensorflow's *softmax* [8] and *mixed precision*, causing numerical instability during the training phase (see **Appendix A** for more details). By applying a fix to this issue, it has been possible to finally train the network smoothly, and, thanks to its halved memory absorption, also to double the training batch size, practically splitting in less than a half the training time, without losing any quality of the final metrics.

In conclusion, the *mixed precision* has been critically important for the development of this work, and it has allowed to create a model that is also more eco-friendly, largely decreasing the computational power required.

5 Results

Considering all the previous tests, it has emerged that the best scoring result can be achieved by doing data augmentation keeping the query in its original language (English), and translating the rest into a different one. In other words, there is not a back translation for the context and the answer, resulting in a multilingual dataset.

Particularly, the best performing experiment has been the so-called *testD1* (see Table 2), which has also been one of the few considerably able to improve over the baseline model. Many others, as depicted in Table 1 and Table 2, stood below the same baseline model, comparing their scores computed on the test set.

Initially, the first mediocre results (*testA1* and *testC1*), as already pointed out, have been used to direct the following experiments toward more promising choices for a better data augmentation. But, interestingly and surprisingly, all the tests following the *testD1* have not only failed to make a further improvement, but they have also shown quite poor scores, compared to those obtained by training the network without any augmentation at all.

Going down deeper into the details, the most interesting point is that the choice of the language has revealed to be more relevant than what initially estimated: repeating the same test (i.e., *testD1* – the best one) using a different source language led to opposite results; despite the new chosen language was closely related to the old one (e.g., French replaced by Spanish).
This fact has brought to a further investigation, pointing toward the hypothesis that French is the only language for which data augmenting the dataset can bring to some benefits, at least with no back translation. Moreover, according to the just mentioned hypothesis, keeping the dataset in a multilingual form should represent the most relevant attempt, since paraphrases are more pronounced, thanks to differences between the two languages. This highlights a subtle weakness in the model, that shows a language-dependent nature. Especially, even though it is language agnostic for what concerns words embeddings (see **Multilingual embedding** section), 96 out of the 396 features, used to encode each word, come from the character embedding layer [2], which, conversely, tries to find out useful relations between characters composing a given word. It is particularly strong in identifying structures such as prefixes and suffixes, enriching the final embedding representation.

Reasoning about the completely different behaviour encountered by using French, Spanish and German has led to a single plausible explanation, based on the following linguistic arguments:

- English and German, despite having the same ancestral origin (both are Germanic languages), show many internal differences, starting from the vocabulary, which has been highly influenced by Latin in the case of English, due to the Roman conquest of Britain [14]. This seems enough to justify the difficulties encountered by the character embedding layer in its work. Moreover, German shows a much higher average length per word [15], which can cause another issue in finding a good embedding at character level, where there are also more truncated words, due to the maximum number of characters allowed for each word (16).
- English and Spanish are highly related, due to the above-mentioned Latin influence for both. In particular, around 30% of English words have a related word in Spanish [16], intended as two words that share a lot of their structure (usually their roots). These words are also called *cognates*.
- Similarly, English is also rich in French *cognates* (mostly the same Latin words shared with Spanish), because both French and Spanish belong to the set of Romance languages.
- Despite the last two points, there is a strong and relevant difference between choosing French or Spanish, at least for what concerns the neural network in discussion: French is the closest language to English [14]. Their similarity is especially represented by their vocabularies, as it has been estimated that roughly 40% of the English words come from French [17]. Furthermore, French has the highest number of *true cognates* (i.e., words that have the same meaning and that are written exactly in the same way) in English: around 1700 [17].
- Historically, in fact, Britain was conquered by Normans in 1066 [14], leading to a strong linguistic contamination (French was exactly the official language for a long medieval period). On the other hand, a language such as Spanish has received other influences from the Arabic language, creating differences that did not allow for many *true cognates* in English.

Summarizing, it has been hypothesized that the strong syntactical and semantic similarities between French and English are the key for the success of the data augmentation, because the character embedding layer is able to detect the same roots as well as the same desinences, and so to discern consistently between one word and another. Contrarily, languages that show closeness in terms of roots, but that are too far in terms of desinences (or generally have a different internal structure), such as Spanish, represent a potential harm for a correct training. Specifically, when using these languages, the model's capabilities to generalize are seriously compromised, also explaining the overfitting encountered in such tests, and highlighting difficulties in defining good embeddings for new words. German behaved similarly to Spanish, because English shares words in common with both languages and, even if it has more *cognates* with Spanish, it also has more *true cognates* with German, due to the old affinity with Germanic languages. Finally, based on the aforementioned grounds, the expectation is that even worse results would have been obtained using languages that do not share anything with English.

Overall, it has been brought to light a potential weakness inside of the architecture, that allows to exploit at best the power of data augmentation with no back translation only to those languages that are extremely close to the target one: an aspect that as theorized before, can be represented by the number of *true cognates* between the two.

6 Conclusions

The main aim of this work was to enhance the QACNNnet [2] in a twofold way: improving previous results and speeding up both training and inference. These two aspects also represent, respectively, the main disadvantage and the main advantage of the model, compared to the state-of-the-art (and more modern) approaches. Indeed, the discussed model is characterized by an extremely lower number of trainable parameters (around 2 millions) with respect to more advanced language models, representing a green AI oriented solution, still capable of reaching very good scores.
In other words, the main goal was to (partially) fill the gap with the best-scoring solutions, while further improving on the efficiency.

After the various tests, we have been able to succeeds in both of the mentioned aims. Particularly, we have improved over the baseline performances by means of a successful operation of data augmentation, based on paraphrases combined with the use of a secondary language (French) and the implementation of a multilingual system. Additionally, the efficiency of the model has been doubled, taking a big step forward by the exploitation of a network that requires half of the memory, thanks to the use of numerical *mixed precision* that allows to halve all the computational costs.

As a final note, we have discovered a potential bias inside of the architecture, that may hinder the strength of a multilingual data augmentation, due to the language dependency of the character embedding layer.
As a future work, we think that it could be interesting to experiment in a new direction where the character embedding job could be assisted by an additional piece of information, encoding the language of the treated words. Thus, it seems reasonable that a language-conditioned version of the model could allow for a better multilingual representation, making a way to reach its full potential through a massive data augmentation.

References

[1] Yu, A.W., et al.: QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension (2018) `https://arxiv.org/abs/1804.09541`

[2] Signorelli G., Sirocchi D.: QACNNnet - Convolutional neural networks and self-attention for question answering `https://github.com/dsr-lab/QACNNnet`

[3] Bojanowski P., et al: Enriching Word Vectors with Subword Information `https://arxiv.org/abs/1607.04606`

[4] Pennington, J., et al.: GloVe: Global Vectors for Word Representation (2014) `https://nlp.stanford.edu/pubs/glove.pdf`

[5] Sze V., et al: Efficient Processing of Deep Neural Networks: A Tutorial and Survey (2017) `https://arxiv.org/pdf/1703.09039.pdf`

[6] Tensorflow's Mixed Precision `https://www.tensorflow.org/guide/mixed_precision`

[7] Rajpurkar, P., et al.: SQuAD: 100,000+ Questions for Machine Comprehension of Text (2016) `https://arxiv.org/abs/1606.05250`

[8] Tensorflow's Softmax `https://github.com/keras-team/keras/blob/v2.11.0/keras/layers/activation/softmax.py#L45-L113`

[9] Argos Translate `https://github.com/argosopentech/argos-translate`

[10] CTranslate2 `https://github.com/OpenNMT/CTranslate2`

[11] Pennington J., et al.: GloVe: Global Vectors for Word Representation (2014) `https://nlp.stanford.edu/pubs/glove.pdf`

[12] FastText (https://fasttext.cc/docs/en/aligned-vectors.html)

[13] Joulin A., et al.:Loss in Translation: Learning Bilingual Word Mapping with a Retrieval Criterion (2018) `https://arxiv.org/pdf/1804.07745.pdf`

[14] Seth Lerer: Inventing English: A Portable History of the Language (2007)

[15] Piantadosi S., et al.: Word lengths are optimized for efficient communication (2011) `https://www.pnas.org/doi/10.1073/pnas.1012551108`

[16] Nash R.: NTC's dictionary of Spanish cognates (1997)

[17] LeBlanc R., et al.: Homographic and non-homographic cognates in English and French (1996)

[18] Tensorflow's MultiHeadAttention `https://github.com/keras-team/keras/blob/v2.11.0/keras/layers/attention/multi_head_attention.py#L130-L726`

Appendix A: *Softmax numerical stability*

As mentioned in the **Computation saving** section, we found that there is a potential issue in the Tensorflow's *Softmax* implementation when coupled with *Mixed precision*. Specifically, the problem is due to numerical instability introduced in the methods found in [8], and reported below:

```
def call(self, inputs, mask=None):
    if mask is not None:
        adder = (1.0 - tf.cast(mask, inputs.dtype)) * (
            _large_compatible_negative(inputs.dtype)
        )
        inputs += adder

    if len(self.axis) > 1:
        return tf.exp(
            inputs
            - tf.reduce_logsumexp(inputs, axis=self.axis, keepdims=True)
        )
    else:
        return backend.softmax(inputs, axis=self.axis[0])
    return backend.softmax(inputs, axis=self.axis)

def _large_compatible_negative(tensor_type):
    if tensor_type == tf.float16:
        return tf.float16.min
    return -1e9
```

Listing 1: Tensorflow's Softmax implementation

Basically, what was taking place in these lines of codes were occasional overflows (involving big negative numbers) when creating the $adder$ variable, resulting in $-inf$ values, eventually transformed in NaN values as soon as the logarithm was computed. Obviously, this was totally ruining the training phase.

The key for solving this problem was to reason about the role of the *adder* variable: it essentially applies a mask by setting a **large negative value** to those parts of the input that must not be considered. The problem is that, when using *mixed precision*, these values are too high in the negative direction, possibly causing dangerous overflows when combined with the inputs. The solution consisted in slightly moving these negative values towards 0, as depicted in the following code:

```
    ...
    adder = (1.0 - tf.cast(mask, inputs.dtype)) * (
        _large_compatible_negative(inputs.dtype) / 2
    )

    ...
```

Listing 2: Tensorflow's Softmax fix

As a final note, we suppose that this Tensorflow's *Softmax* implementation issue was critical in our model mainly due to its depth, and it's negative effect should be less significant for shallower models. Moreover, this problem is even more noticeable due to the usage of the Tensorflow's *MultiHeadAttention* [18], where we actually applied the fix.