# Title

Hierarchical Attention Networks for Document Classification

# Author

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, Eduard Hovy

# Abstract

Author proposes **a hierarchical attention network for document classification**. It has 2 distinctive characteristics: (i) it has a **hierarchical structure** that mirrors the hierarchical structure of documents; (ii) it has **two levels of attention mechanisms applied at the word and sentence-level**, enabling it to attend differentially to more and less important content when constructing the document representation.

# Issue

Traditional approaches of text classification represent documents with **sparse lexical features**, such as n-grams, and then use a linear model or kernel methods on this representation. More recent approaches used deep learning, such as convolutional neural networks and recurrent neural networks based on long short-term memory (LSTM) to learn **text representations**. They all ignore the knowledge of **hierarchical document structure** and **different importance of words and sentences**.

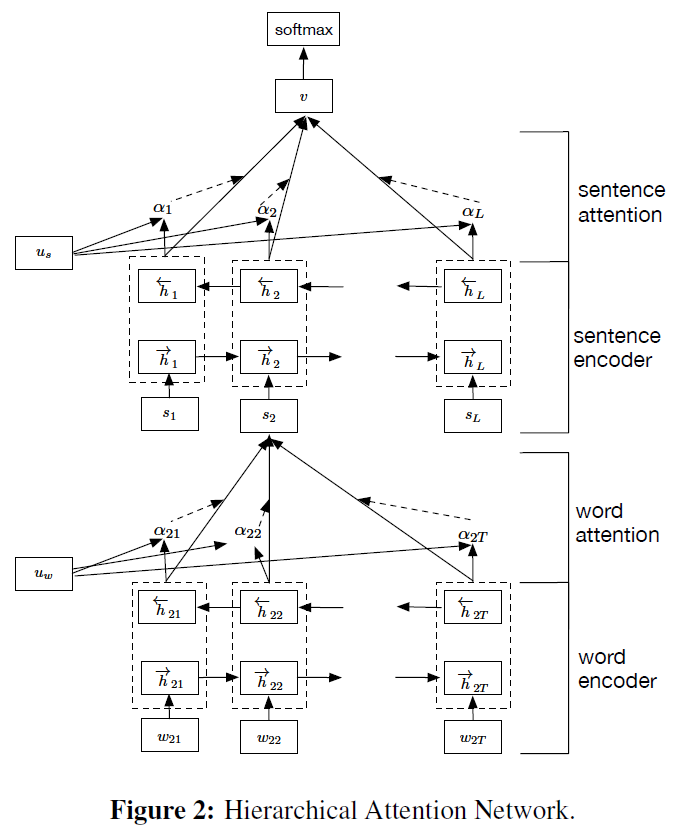
# Method

## GRU-based sequence encoder

The GRU uses a gating mechanism to track the state of sequences without using separate memory cells. There are two types of gates: the reset gate and the update gate . They together control how information is updated to the state. At time t, the GRU computes the new state as

This is a linear interpolation between the previous state and the current new state computed with new sequence information. The gate decides how much past information is kept and how much new information is added. is updated as:

where is the sequence vector at time t. The candidate state is computed in a way similar to a traditional recurrent neural network (RNN):

Here is the reset gate which controls how much the past state contributes to the candidate state. If rt is zero, then it forgets the previous state. The reset gate is updated as follows:

## Hierarchical Attention

### Word Encoder

We obtain an annotation for a given word by concatenating the forward hidden state and backward hidden state , i.e., , which summarizes the information of the whole sentence centered around :

where is word embedding.

### Word Attention

That is, we first feed the word annotation hit through a one-layer MLP to get as a hidden representation of , then we measure the importance of the word as the similarity of with a word level context vector and get a normalized importance weight through a softmax function. After that, we compute the sentence vector (we abuse the notation here) as a weighted sum of the word annotations based on the weights. The context vector can be seen as a high level representation of a fixed query “what is the informative word” over the words like that used in memory networks. The word context vector is randomly initialized and jointly learned during the training process.

### Sentence Encoder

Given the sentence vectors , we can get a document vector in a similar way. We use a bidirectional GRU to encode the sentences:

We concatenate and to get an annotation of sentence , i.e., . summarizes the neighbor sentences around sentence but still focus on sentence .

### Sentence Attention

where is the document vector that summarizes all the information of sentences in a document. Similarly, the sentence level context vector can be randomly initialized and jointly learned during the training process.

### Document Classification

The document vector v is a high level representation of the document and can be used as features for document classification:

We use the negative log likelihood of the correct labels as training loss:

where is the label of document .