The goal is to create word form representations that meet the following two properties:

- They should account for phenomena arising from orthographic similarity between words such as stability, edge effects, transposed letter effects, and relative position effects.
- They should be arranged in a low dimensional space such that distance between any two words indicate their orthographic similarity.

We consider two approaches: spatial coding and alphabetic representation. Spatial coding meets the first property in that effects like transposed letter, stability and relative positions are accounted for. However, there is no sense of distance between two word represenations in such a space, which makes it difficult to explore statistics and carry out tasks like word clustering. In alphabetic representation (for 3 letter words), each word is assigned a 3D point based on the three alphabets that form the word. Although there is a sense of distance between 3D points, such a represenation does not meet the first property. For example, words *dog* and *dig* and words *dog* and *dug* should be equally similar, but are not. Furthermore, it is challenging to represent variable length words in such a space.

The approach proposed here is to learn a 3D space representation by iterating over orthographic similarities between words. *The motivating idea is that the transitive property holds for words' orthographic similarity.* That is, if w_1 is similar to w_2 and w_2 is similar to w_3, then w_1 is also similar to w_3.

Consider a training sequence of L words given by W. Let f is an operator that determines orthographic similarity between words (such operators are widely available in literature). Also let N is the dimension of the transformed representation (assumed to be 3 for our discussion). **The algorithm works as follows:**

- STEP 1: Place the first word at the origin of N dimensional space.
- STEP 2: For every new word w, compute the orthographic similarity with last N words and locate the position of w by satisfying the whole euclidean distance constraints.

$$\sqrt{(x_w - x_{w-1})^2 + (y_w - y_{w-1})^2 + (z_w - z_{w-1})^2} = f(w, w-1) \tag{1}$$

$$\sqrt{(x_w - x_{w-2})^2 + (y_w - y_{w-2})^2 + (z_w - z_{w-2})^2} = f(w, w-2) \tag{2}$$

$$\sqrt{(x_w - x_{w-3})^2 + (y_w - y_{w-3})^2 + (z_w - z_{w-3})^2} = f(w, w-3) \tag{3}$$

Here, x_w, y_w and z_w are the three unknowns which can be determined with three equations. Note that there are N equations for an N-dimensional space.

- STEP 3: This step is applied prior to step 2. The last N words picked should be most similar. Here, similarity is defined as the number of same letters in the two words.

After repeating the process for the complete training sequence, the whole lexicon can be represented in a low dimensional space that meets the two properties defined at the beginning of the document.