show that our approach outperforms other unsupervised state-of-the-art summarizers.

2 Methodology

The proposed summarization methodology, denoted by Gamp[1], is based on the MDL principle that is defined formally as follows (Mitchell, 1997). Given a set of models $\mathcal{M}$, a model $M \in \mathcal{M}$ is considered the *best* if it minimizes $L(M) + L(D|M)$ where $L(M)$ is the bit length of the description of M and $L(D|M)$ is the bit length of the dataset D encoded with M.

In our approach, we first represent an input text as a transactional dataset. Then, using the Krimp dataset compression algorithm (Vreeken et al., 2011), we build the MDL for this dataset using its frequent sequential itemsets (word sequences). The sentences that cover most frequent word sequences are chosen to a summary. The following subsections describe our methodology in detail.

2.1 Problem statement

We are given a single text or a collection of texts about the same topic, composed of a set of sentences $S_1, \ldots, S_n$ over terms $t_1, \ldots, t_m$. The word limit W is defined for the final summary.

We represent a text as a *sequential transactional dataset*. Such a dataset consists of *transactions* (sentences), denoted by $T_1, \ldots, T_n$, and unique items (terms[2]) $I_1, \ldots, I_m$. Items are unique across the entire dataset. The number n of transactions is called the *size* of a dataset. Transaction T_i is a sequence of items from $I_1, \ldots, I_m$, denoted by $T_i = (I_{i_1}, \ldots, I_{i_k})$; the same item may appear in different places within the same transaction. *Support* of an item sequence s in the dataset is the ratio of transactions containing it as a subsequence to the dataset size n, i.e., $supp(s) = \frac{|\{T \in D | s \subseteq T\}|}{n}$. Given a support bound $Supp \in [0, 1]$, a sequence s is called *frequent* if $supp(s) \geq Supp$.

According to the MDL principle, we are interested in the minimal size of a compressed dataset $D|CT$ after frequent sequences in D are encoded with the compressing set–codes from the *Coding Table* CT, where shorter codes are assigned to more frequent sequences. The description length of non-encoded terms is assumed proportional to their length (number of characters). We rank sentences by their coverage of the best compressing set, which is the number of CT members in the sentence. The sentences with the highest coverage score are added to the summary as long as its length does not exceed W.

2.2 Krimping text

The purpose of the Krimp algorithm (Vreeken et al., 2011) is to use frequent sets (or sequences) to compress a transactional database in order to achieve MDL for that database. Let $FreqSeq$ be the set of all frequent sequences in the database. A collection CT of sequences from $FreqSeq$ (called the *Coding Table*) is called *best* when it minimizes $L(CT) + L(D|CT)$. We are interested in both *exact* and *inexact* sequences, allowing a sequence to have gaps inside it as long as the ratio of sequence length to sequence length with gaps does not exceed a pre-set parameter $Gap \in (0, 1]$. Sequences with gaps make sense in text data, as phrasing of the same fact or entity in different sentences may differ. In order to encode the database, every member $s \in CT$ is associated with its binary *prefix code* c (such as Huffman codes for 4 members: 0, 10, 110, 111), so that the most frequent code has the shortest length. We use an upper bound C on the number of encoded sequences in the coding table CT, in order to limit document compression. Krimp-based extractive summarization (see Algorithm 1) is given a document D with n sentences and m unique terms. The algorithm parameters are described in Table 1:

#	*note*	*description*	*affects*
1	**W**	summary words limit	summary length
2	**Supp**	minimal support bound – minimal fraction of sentences containing a frequent sequence	number of frequent word sequences $\|FreqSeq\|$, compression rate
3	**C**	maximal number of codes	as in 2
4	**Gap**	maximal allowed sequence gap ratio	as in 2

Table 1: Parameters of Algorithm 1.

The algorithm consists of the following steps.

1. We find all frequent term sequences in the document using Apriori-TID algorithm (R and R, 1994) for the given $Supp$ and Gap and store them in set $FreqSeq$, which is kept in Standard Candidate Order[3]. The coding table CT is initialized to contain all single normalized terms and their frequencies. CT is always kept in Standard Cover Order[4] (Steps

[1] abbreviation of two words: GAp and kriMP

[2] normalized words following tokenization, stemming, and stop-word removal

[3] first, sorted by increasing support, then by decreasing sequence length, then lexicographically

[4] first, sorted by decreasing sequence length, then by decreasing support, and finally, in lexicographical order