Task

- ✓ Prove M is Maximum iff no augment path
- ✓ Explain
- ✓ Show work

Assume this is true:

An augmented path is one that begins on a free vertice and ends on a different free vertice, traversing alternating edges that are and aren't in the matching set M, starting with an edge not in the set. (The second half of that is implied from the definition of a free vertex).

M : a matching M in a graph is a set of pairwise non-adjust edges. No edges share a common vertex.

Maximum: M that contains the largest possible number of edges

Prove that a matching M is maximum iff there is no augmenting path in G.

Assume M is a matching set in graph G.

Suppose there is an augmenting path in G named P. This implies that P begins and ends on free vertices, alternating between edges in the set M and not in the set M. This means that if you take the set of edges in P that are not part of the set M, you can create a new matching set P'. P' will have more edges than originally the matching set M. This proves there does exist a matching set with a larger number of edges than in M. This by definition contradicts that M could be Maximum as M would need to contain the largest possible number of edges, but we've shown that P' has more edges. This proves that if there is an augmenting path in G given the matching set M, then M is not the maximum as the matching set P' is possible.

Thus we can conclude that if there is no augmenting path, then M must be maximum.

```
public LatentSemanticAnalysis(List<List<String>> documents, int k)
	{
		vs_model = new VectorSpaceModel();
		//List<Term[]> list = vs_model.toTFIDFs(documents);
		
		List<Term[]> list = new ArrayList<>();
		for (List<String> document : documents)
		list.add(vs_model.toBagOfWords(document, true));
		
		int T = vs_model.getTermSize(), D = list.size();
		td_matrix = new Basic2DMatrix(T, D);
		
		for (int docID=0; docID<D; docID++)
			for (Term term : list.get(docID))
				td_matrix.set(term.getID(), docID, term.getScore());
		
		////toLSA(k);
	}


public List<ObjectDoublePair<String>> getTopSimilarTerms(String term, int k)
	{
		
		List<ObjectDoublePair<String>> list = new ArrayList<>();
		int termID= vs_model.getID(term);
		if(termID < 0) return list;
		
		for(int i = td_matrix.rows()-1; i>=0;i--)
		{
			if((i)!=termID)
				list.add(new ObjectDoublePair<String> (
vs_model.getTerm(i),

	getCosineSimilarityTerm(termID,i)));
		}
```

```
        Collections.sort(list, Collections.reverseOrder());
        return (k> list.size())? list : list.subList(0, k);
}
```