Design a system to determine the 10 words that best represent the text content of a HTML page. The system should be flexible enough to handle content in any natural language.

1. Unlike plain texts, HTML has tags, which gives a clue of the importance of different words. An HTML page may contain different levels of headings (e.g. <h1>, <h2>), various emphasizers (e.g. <b>, ,). We can also identify words with different font color, size or font family from the majority. When there is some image or video, those words aligned to the image seem more important.
2. 2.If we have access to a bunch of HTML pages, we could make use of inverse document frequency combined with the features mentioned in the above paragraph.
   For each word in our target HTML page, we could determine whether this word appears in other pages, with a weight heuristically determined according to their tags. The more important the tag is , the more weight we shall allocate. Then we have the weighted document frequency df for a particular word in the target HTML. Let N be the total HTML pages in the collection, then we may sort each word by tf-idf scores (tf-idf = (term frequency) * log(N/df)). Return the top ten.

As the definition of a keyword is unclear, right now, I can't give a rule of thumb to determine weights for different tags in the HTML.

Design a "smart filing" system for Evernote that can learn to anticipate the notebook and/or tags the user wants to assign to new content.

It's really a complicated problem. Using a multi-class classification method is not sufficient as it has a fixed and large tag space. Worse still, it can not determine the ordering of different tags. I think we can decompose the problem into two steps. In the first step, we create tag candidates. In the second step, we rank the tags according to some algorithm and yield tags with highest rankings. (Assume we have access to a large number of tagged contents for training and we have ordering of tags given to one document according to preference.)

There are several ways to generate candidates:

1. Tags already used by the user
2. If the text is excerpted from some article and the text has been used by other users, include the tags most people use for this text.
3. If the text is written by the user, we can extract keywords from the title or content according to the first problem
4. To generate more tags, we can find similar texts by clustering algorithms, say KNN and include the tags used by similar texts. But this maybe costly. We first need to convert the text to a vector, say using tf-idf scores as vectors. And then we can use KNN.

Once we have the candidates, we set up a ranking model and convert tags into feature vectors

1. The last time the user used the tag, and how many times the user have used the tag if the tag has been used before.
2. The lexical category of tags. In my personal experience, I tend to use noun as tags.
3. tf - idf scores

Then we can build ranking models by several machine learning techniques like Ranking SVM.