Task 1:

Intuitively, the weights of the fields I experimented are inversely proportional to the length of the fields and directly proportional to how the world perceive the content of the document (eg: anchor texts). Here is a subset of the weights I experimented with:

ndcgTrain	ndcgDev	ndcgDiff	bodyweight	anchorweight	headerweight	titleweight	urlweight
0.85798	0.83277	0.02521	1	3	1	1	1
0.854627	0.831727	0.0229	1	3	1	1	2
0.854714	0.830567	0.024147	1	3	1	2	1
0.854698	0.83043	0.024268	1	3	1	2	3
0.854877	0.829747	0.025131	1	5	1	3	3
0.854638	0.829082	0.025556	1	5	5	2	4
0.85519	0.829069	0.026121	1	5	3	3	4
0.854741	0.828547	0.026194	1	5	1	1	4
0.855455	0.829155	0.0263	1	5	3	1	4
0.846234	0.831697	0.014537	1	1	5	1	1

From the data above, I choose the weights shown below as they give high NDCG scores of the training data with a relatively low difference in NDCG scores of the test data.

task1_W_body = 1
task1_W_anchor = 3
task1_W_header = 1
task1_W_title = 1
task1_W_url = 1

I have also sub-linearly scaled the document and query term frequencies as the ranking of a document to a query is not linear to the term frequencies. Scaling them sub-linearly makes the term frequencies contribute highly at the start and then they plateau later on.

Task 2:

I based the weights assigned to the various fields on the data collected for task 1. I experimented with different combinations of the normalization factors and K1 value and choose the values which gave high NDCG scores of the training data with a relatively low difference in NDCG scores of the test data.

ndcgTrain	ndcgDev	ndcgDiff	k1	bbody	banchor	bheader	btitle	burl
0.868413	0.851629	0.016784	6	0.75	0.75	0.75	1	0.75
0.868443	0.851557	0.016886	6	0.75	0.25	0.75	0.75	0.5
0.868292	0.848599	0.019693	5	0.75	0.5	0.75	1	0.25
0.868438	0.847909	0.020529	5	0.75	0.25	0.25	1	1
0.870433	0.85512	0.015313	10	0.75	0.25	0.75	1	0.75
0.871037	0.854066	0.01697	10	0.75	0.5	0.25	0.75	0.75
0.869294	0.851035	0.018259	9	0.75	0.25	0.75	1	0.75
0.870648	0.851499	0.019149	9	0.75	0.25	0.5	1	0.5
0.870266	0.85168	0.018587	8	0.75	0.25	0.75	1	0.75
0.868862	0.849583	0.019279	8	0.75	0.5	0.25	0.5	0.75

The initial increments in the pagerank contributes more to the score when compared to the final increments. This property is better captured by representing it on a logarithmic scale when compared to using a sigmoid or saturation curve.

Here is a subset of the weights I experimented with:

ndcgTrain	ndcgDev	ndcgDiff	lambda	lambda'
0.88052	0.86082	0.0197	8	9
0.879834	0.85877	0.021064	8	8
0.878304	0.858835	0.019469	8	7
0.877913	0.858314	0.019599	10	10
0.877257	0.857541	0.019715	5	10
0.877669	0.857846	0.019823	9	9
0.87764	0.857803	0.019838	5	8
0.877298	0.857273	0.020026	4	8
0.877697	0.857664	0.020032	6	5
0.877981	0.857712	0.020269	7	6
0.879199	0.858913	0.020285	10	8

From the data above, the weights I choose are:
Task2_W_body = 1
Task2_W_anchor = 3
Task2_W_header = 1
Task2_W_title = 1
Task2_W_url = 1
Task2_B_body = 0.75
Task2_B_anchor = 0.25
Task2_B_header = 0.75
Task2_B_title = 1
Task2_B_url = 0.75
Task2_K1 = 10
Task2_lambda = 8
Task2_ lambda' = 9

Task 3:

The smallest window scorer is extended from the cosine similarity scorer. I based the weights assigned to the various fields on the data collected for task 1.

The boost I gave to the score is inversely proportional to the window size with a maximum boost given when the query words are found successively next to each other.

ndcgTrain	ndcgDev	ndcgDiff	B
0.862045	0.83629	0.025755	1
0.859442	0.83218	0.027263	2
0.857907	0.828635	0.029272	4
0.857429	0.826568	0.030861	5
0.857325	0.830364	0.026961	3
0.856291	0.824649	0.031642	6
0.856017	0.823096	0.032921	7
0.854162	0.823996	0.030166	8
0.853677	0.82362	0.030058	9
0.853087	0.82389	0.029197	10

The weights is chose for the task are:

task3_W_body = 1
task3_W_anchor = 3
task3_W_header = 1
task3_W_title = 1
task3_W_url = 1
task3_B = 1

Extra:

I have extended the smallest window scorer to stem both the document and query words, using porter stemmer, before proceeding with vectorization. Also, I gave a weighted boost based on the field where the smallest window is found.

Here is a subset of the weights I experimented with:

ndcgTrain	ndcgDev	ndcgDiff	titleWindow	bodyWindow	anchorWindow	headerWindow
0.86975	0.843528	0.026222	3	1	4	3
0.865841	0.846532	0.019309	3	1	3	2
0.865841	0.846532	0.019309	3	1	3	4
0.865639	0.847678	0.017961	2	1	3	3
0.863581	0.846299	0.017282	4	1	2	3
0.862432	0.84438	0.018051	3	2	4	3
0.860943	0.842954	0.01799	3	2	3	2
0.860943	0.842954	0.01799	3	2	3	4
0.860032	0.844344	0.015688	2	2	3	3
0.859381	0.841597	0.017784	4	2	2	3

The weights is chose for the task are:
extra_titleWindowWeight = 3
extra_bodyWindowWeight = 1
extra_anchorWindowWeight = 4
extra_headerWindowWeight = 3

Other metrics:

One query-independent metric that can be incorporated is to come up with a subset of the subdomains of Stanford.edu which, in general, can be treated as an authoritative list which is scored higher that its counterparts.

One can also use a thesaurus to search for words with similar meaning as the query words.