This is the last version I just finished. I’d like to make a summary of my current implementation.

The features of this version include:

1. Searching and highlighting the entities in the input area. User can type or paste text in the input area, the entities can be identified and highlighted automatically in real time.
2. When mouse hover on the highlight text, a DIV with the description and image of the entity will show on the right side. Since some images’ URLs are invalid, so the DIV will show the image only if it can be loaded successfully.
3. When use clicks on the identified entity, a new browser tab will jump to the URL defined in the entity data set.
4. The entity searching is case insensitive.

I had tested these features in Chrome 43, Firefox 36 and IE11 on Windows 8.1. Both in Chrome and Firefox works fine, but in IE sometimes text overlaps. I don't have Mac, so I didn’t test it in Safari.

Some implement details:

1. I didn’t add any backend code, but loaded all entities in a js file. Since we only have 4000 entities, the text searching is still fast.
2. My old implementation used a DIV with the property value of “contenteditable” as True. The problem of this solution is when the text rendered or pasted, the cursor may move to random position. My current solution is putting a textarea under a DIV, user can type and paste text in the textarea, the rendered text, which includes some spans with style red color, will show in the front DIV. The most tricky part of this solution is to enable the under layer textarea receiving user input, I had to disable the all the events of the front DIV, but we also need the front DIV can sense the mouse over and click event. The solution to this problem is: When any highlight span created, its reference will be saved in an array. When mouse event fired, the program will check if mouse is in the dimension of any span, and if it is, the information of the entity the span highlighted will show.
3. The other tricky problem is entity search. One issue is some entity title is just a letter, like “W”. The words include this letter, like “Who, When” should not be identified as entity. The other is some titles are the part of others, like “United States” is part of “United States Secretary of the Treasury”. The longer title should not be identified as the shorter ones. The solution to these problem are: First, sort all the titles by length, so the longer title will be searched at first. Then when any title is found in text, the position range, start and end position of the title in the text, will be saved. When a new title is found in the text, the new found range will be checked in the saved ranges. If it is in other ranges, which means this part of text is belong to some longer title, it should not be rendered anymore. The program also check the pre and post character of the found title, make sure it is a whole word, not part of other word.
4. My old design of description panel is put it under the highlight text. But I found some times the panel is too high, user cannot see all of it whether I put it above or under the highlight text. So I move it to the right of the page. User can always see it, and it will never overlap the text.
5. Other issues I’d worked on includes: Browser compatibility, automatically adjust the position and size of elements, text wrap, line break style etc. I originally used jQuery and Bootstrap, but later I found some operations are really complex, and the frameworks effected my code, so I only use pure JavaScript instead. There should some problems I hadn’t resolved for the time reason. But I believe it’s enough to demonstrate my skill.

How to scale?

If there are millions of entities, we should use server side to do search job. The client JavaScript should send the text to the server side, the back-end application will return the found titles and ranges to the client. We also can speed up the searching by parallel computing, like map-reduce. We can map search task to search servers, each search sever may search part of entities, and the search results will be summarized in the reduce server. The other idea is to combine both server side and client side searching. In the client side only frequent entities will be searched and rendered in real time, the server side will search all the entities and return the result periodically, this solution will reduce the server side pressure, avoid sending searching request when any small change happened.