problem 1.

(a)

$d(\text{dog}, \text{animal}) = \sqrt{2^2+1^2+1^2+2^2+1^2} = \sqrt{11}$

$d(\text{cat}, \text{animal}) = \sqrt{2^2+3^2+3^2+7^2} = \sqrt{71}$

$d(\text{computer}, \text{animal}) = \sqrt{2^2+3^2+2^2+2^2} = \sqrt{21}$

$d(\text{run}, \text{animal}) = \sqrt{2^2+5^2+3^2+3^2+1^2} = \sqrt{48}$

$d(\text{mouse}, \text{animal}) = \sqrt{7^2+5^2+1^2+3^2+3^2} = \sqrt{93}$

most similar to 'animal': "dog".

$$\text{simcos}(\text{dog}, \text{animal}) = \frac{12+12+6}{\sqrt{4^2+4^2+1^2+1^2}\cdot\sqrt{2^2+3^2+3^2+3^2}} = \frac{30}{\sqrt{90}\cdot\sqrt{31}} = 0.85$$

$$\text{simcos}(\text{cat}, \text{animal}) = \frac{8+9+30}{\sqrt{4^2+3^2+3^2+10^2}\cdot\sqrt{2^2+3^2+3^2+3^2}} = \frac{47}{\sqrt{134}\sqrt{31}} = 0.73$$

$$\text{simcos}(\text{computer}, \text{animal}) = \frac{15+15}{\sqrt{5^2+5^2}\cdot\sqrt{31}} = 0.76$$

$$sim_{cos}(run, animal) = \frac{8+9+12}{\sqrt{9^2+3^2+5^2+3^2+9^2}\cdot\sqrt{31}}$$

$$= 0.60$$

$$sim_{cos}(mouse, animal) = \frac{9+30+12}{\sqrt{31}\ \sqrt{2^2+10^2+5^2+9^2+3^2}}$$

$$= 0.67$$

most similar to animal → 'dog'.

problem 2.

we can check if there are overlap between the two synsets, and we can also check any synset.hyponyms() has overlap with words in another synsets.

Programming component

Part 1: candidate synonyms from wordnet

```
def get_candidates(lemma, pos):
    # Part 1
    possible_synonyms = []
    l1=wn.lemmas(lemma,pos=pos)
    for l in l1:
        lex=l.synset().lemmas()
        for le in lex:
            word=le.name()
            if word not in possible_synonyms and word !=lemma:
                if '_' in word:
                    word.replace('_',' ')
                possible_synonyms.append(word)
    return possible_synonyms
```

part 2: wordnet frequency baseline

```
def wn_frequency_predictor(context):
    lemma=context.lemma
    pos=context.pos
    l1=wn.lemmas(lemma,pos=pos)
    record_req={}
    for l in l1:
        lex=l.synset().lemmas()
        for le in lex:
            word=le.name()
            if word != lemma:
                if word not in record_req:
                    record_req[word]=le.count()
                else:
                    record_req[word]+=le.count()
    return max(record_req,key=record_req.get)
```

part 3:simple lesk algorithm

```
def wn_simple_lesk_predictor(context):
    lemma=context.lemma
    pos=context.pos
    total_sentence=set(context.left_context+context.right_context)
    sentence=[]
    def_reference={}
    for word in total_sentence:
        if word not in stopwords.words('english'):
```

```
            sentence.append(PorterStemmer().stem(word))
    sentence=set(sentence)
    l1=wn.lemmas(lemma,pos=pos)
    for l in l1:
        lex=l.synset().lemmas()
        for le in lex:
            s=le.synset()
            definition=word_tokenize(s.definition())+s.examples()
            for hy in s.hypernyms():
                definition+=word_tokenize(hy.definition())
                for ex in hy.examples():
                    definition+=word_tokenize(ex)
            definition=set([WordNetLemmatizer().lemmatize(word) for word in definition
if word not in stopwords.words('english')])
            overlap_count=len(definition.intersection(sentence))
            if overlap_count:
                word=le.name()
                if word !=lemma:
                    if word not in def_reference:
                        def_reference[word]=overlap_count
                    else:
                        def_reference[word]+=overlap_count
    if not def_reference:
        return wn_frequency_predictor(context)
    else:
        return max(def_reference,key=def_reference.get)
```

part 4: most similar synonym

```
def predict_nearest(self,context):
    possible_synonyms=get_candidates(context.lemma,context.pos)
    max_sim=0
    ans=None
    for sy in possible_synonyms:
        if sy in self.model.wv.vocab:
            this_sim=self.model.similarity(context.lemma,sy)
            if this_sim>max_sim:
                max_sim=this_sim
                ans=sy
    return ans
```

part 5: context and word embedding:

```
def predict_nearest_with_context(self, context):
    total_sentence = set(context.left_context[-5:] + context.right_context[0:5])
    sentence = []
```

```
    vector_sum=self.model.wv[context.lemma]
    for word in total_sentence:
        if word not in stopwords.words('english'):
            sentence.append(word)
    for word in sentence:
        if word in self.model.wv.vocab:
            vector_sum=vector_sum+self.model.wv[word]
    possible_synonyms=get_candidates(context.lemma,context.pos)
    max_sim=0
    ans=None
    for sy in possible_synonyms:
        if sy in self.model.wv.vocab:
            this_sim=cos(self.model.wv[sy],vector_sum)
            if this_sim>max_sim:
                max_sim=this_sim
                ans=sy
    return ans
```

part6:

```
def own_predict_nearest_with_context(self, context):
    total_sentence = set(context.left_context + context.right_context)
    sentence = []
    vector_sum=self.model.wv[context.lemma]
    for word in total_sentence:
        if word not in stopwords.words('english'):
            sentence.append(word)
    for word in sentence:
        if word in self.model.wv.vocab:
            vector_sum=vector_sum+self.model.wv[word]
    possible_synonyms=get_candidates(context.lemma,context.pos)
    max_sim=0
    ans=None
    for sy in possible_synonyms:
        if sy in self.model.wv.vocab:
            this_sim=cos(self.model.wv[sy],vector_sum)
            if this_sim>max_sim:
                max_sim=this_sim
                ans=sy

    return ans
```